

ONLINE ARABIC HANDWRITTEN TEXT RECOGNITION USING SYNTACTICAL-BASED TECHNIQUES

BY

DHIAA ABDULRAB ALI MUSLEH

A Dissertation Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

In

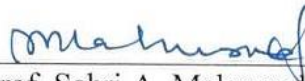
COMPUTER SCIENCE AND ENGINEERING

May 2016

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN- 31261, SAUDI ARABIA
DEANSHIP OF GRADUATE STUDIES

This thesis, written by **DHIAA ABDULRAB ALI MUSLEH** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING.**

Dissertation Committee



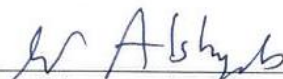
Prof. Sabri A. Mahmoud
(Advisor)



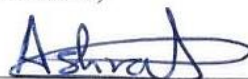
Prof. Radwan Abdel Aal
(Member)



Prof. Moustafa Elshafei
(Member)




Dr. Mohammad Alshayeb
(Member)



Dr. Ashraf Mahmoud
(Member)



Dr. Adel F. Ahmed
Department Chairman


Dr. Salam A. Zummo
Dean of Graduate Studies



9/6/16
Date

© Dhiaa Musleh
2016

Dedicated to my mother and father

ACKNOWLEDGMENTS

First and foremost thanks to Allah who gave me the strength, the patience and the ability to accomplish this work. Peace and blessing of Allah be upon his last messenger Mohammed, who guided us to the right path.

Acknowledgment is due to King Fahd University of Petroleum and Minerals for giving me this research opportunity and supporting me throughout my graduate studies.

I would like to express my deep appreciation and gratitude to my thesis advisor Prof. Sabri Mahmoud for his continuous help, guidance, encouragement and invaluable support. I also wish to extend my thanks to my thesis committee members, Prof. Radwan Abdel Aal, Prof. Moustafa Elshafei, Dr. Mohammad Alshayeb and Dr. Ashraf Mahmoud for their suggestions and comments.

I also would like to thank my colleagues for their support, motivation and encouragement. Special thanks to my brothers Dr. Fahd Al-Haidari and Dr. Rashad Othman.

Special and deep thanks to my wife and my children who always support me with their love, patience, encouragement and constant prayers.

I wish to express my love and gratitude to my beloved parents, brothers and sisters for their endless love and motivation.

Finally, I extend my thanks to everyone who helped directly or indirectly to get this work done.

TABLE OF CONTENTS

| | |
|---|------|
| ACKNOWLEDGMENTS | III |
| TABLE OF CONTENTS..... | IV |
| LIST OF TABLES | IX |
| LIST OF FIGURES | X |
| ABSTRACT..... | XIII |
| ملخص الرسالة..... | XV |
| CHAPTER 1 INTRODUCTION | 1 |
| 1.1 Handwriting Recognition..... | 2 |
| 1.2 Offline vs. Online Handwriting Recognition | 2 |
| 1.3 Challenges in Arabic online handwriting recognition..... | 3 |
| 1.4 Motivation..... | 4 |
| 1.5 Contributions of the Thesis | 4 |
| 1.6 Organization of this Thesis | 8 |
| CHAPTER 2 LITERATURE REVIEW | 9 |
| 2.1 CHARACTERISTICS OF ARABIC SCRIPT..... | 9 |
| 2.2 General model for Arabic online text recognition system | 13 |
| 2.3 Arabic online handwritten databases | 14 |
| 2.4 Preprocessing techniques | 17 |
| 2.4.1 Resampling and noise removal | 19 |
| 2.4.2 Baseline Detection | 22 |
| 2.5 Segmentation techniques | 24 |
| 2.6 Feature extraction techniques | 26 |
| 2.6.1 Structural Features | 26 |

| | | |
|---|---|-----------|
| 2.6.2 | Statistical Features | 30 |
| 2.7 | Classification approaches | 35 |
| CHAPTER 3 FUZZY MODELING FOR ARABIC ONLINE DIGITS RECOGNITION..... | | 43 |
| 3.1 | Introduction..... | 44 |
| 3.2 | Preprocessing..... | 47 |
| 3.3 | Feature Extraction..... | 48 |
| 3.3.1 | Shape Features | 48 |
| 3.3.2 | Directional Features | 49 |
| 3.3.3 | Histogram-based Features..... | 49 |
| 3.4 | Automatic Fuzzy Models' Generation..... | 51 |
| 3.4.1 | Automatic Fuzzy Modeling..... | 51 |
| 3.4.2 | Model Trend Line (MTL) | 52 |
| 3.4.3 | Model Envelop (ME) | 53 |
| 3.5 | Classification | 55 |
| 3.5.1 | Zero/Nonzero Classification | 56 |
| 3.5.2 | Fuzzy Classification..... | 56 |
| 3.6 | Experimental Work..... | 60 |
| 3.7 | Conclusions..... | 64 |
| CHAPTER 4 ONLINE ARABIC CHARACTER RECOGNITION BASED ON GRAPHEME MODELING | | 66 |
| 4.1 | Introduction..... | 66 |
| 4.2 | Collected Data | 69 |
| 4.3 | Preprocessing steps | 70 |
| 4.4 | Graphemes extraction | 71 |
| 4.5 | Feature Extraction..... | 72 |
| 4.5.1 | Writing Direction Feature | 72 |
| 4.5.2 | Orientation Histogram-based features..... | 73 |

| | | |
|--|---|----|
| 4.5.3 | Polar Angular features | 73 |
| 4.5.4 | Chain Code features | 74 |
| 4.5.5 | Curliness features | 75 |
| 4.5.6 | Loop detection feature | 76 |
| 4.5.7 | Other features | 76 |
| 4.6 | Graphemes' Codebook Generation | 77 |
| 4.6.1 | Graphemes Clustering | 77 |
| 4.6.2 | Codebook Generation | 77 |
| 4.6.3 | Codebook Reduction | 78 |
| 4.7 | Experimental work | 79 |
| 4.7.1 | Grapheme Classification | 79 |
| 4.7.1.1 | Evaluating the generated classes | 79 |
| 4.7.1.2 | Evaluating the reduction of the codebook using Bag-of-classes | 81 |
| 4.7.2 | Character Classification | 83 |
| 4.7.2.1 | Characters' representation | 84 |
| 4.7.2.2 | Used Classifiers | 86 |
| 4.7.2.3 | Experimental Results | 86 |
| 4.7.3 | Conclusions | 89 |
| CHAPTER 5 ARABIC ONLINE TEXT RECOGNITION | | 91 |
| 5.1 | Overview of the Arabic Online Text Recognition Process | 91 |
| 5.2 | Training phase | 92 |
| 5.2.1 | Preprocessing | 94 |
| 5.2.2 | Grapheme Extraction | 96 |
| 5.2.3 | Feature Extraction | 98 |
| 5.2.3.1 | Angles Quantization Feature | 98 |
| 5.2.3.2 | Orientation Histogram-based Feature | 98 |
| 5.2.3.3 | Curliness feature | 99 |

| | | |
|---------|---|-----|
| 5.2.3.4 | Curvature and Writing Direction Feature | 99 |
| 5.2.3.5 | Polar Angular feature | 100 |
| 5.2.3.6 | End-Points Feature | 100 |
| 5.2.3.7 | Loop detection feature | 101 |
| 5.2.3.8 | Shape features | 101 |
| 5.2.4 | Graphemes' Codebook Generation | 101 |
| 5.2.5 | Feature Selection Approach | 103 |
| 5.2.6 | Codebook improvement based on the selected features | 103 |
| 5.2.7 | Fuzzy Modeling (generating graphemes' fuzzy models) | 104 |
| 5.3 | Recognition phase | 105 |
| 5.3.1 | Preprocessing | 105 |
| 5.3.2 | Baseline Estimation | 106 |
| 5.3.3 | Delayed strokes handling | 107 |
| 5.3.4 | Graphemes' Extraction | 110 |
| 5.3.4.1 | Loop detection | 110 |
| 5.3.4.2 | Open loop detection | 112 |
| 5.3.4.3 | Finding Possible Segmentation Points (PSP) | 113 |
| 5.3.4.4 | Filtering Possible Segmentation Points (PSPs) | 113 |
| 5.3.5 | Feature Extraction | 114 |
| 5.3.6 | The graphemes' relative height feature (Grapheme/Line height ratio) | 115 |
| 5.4 | Arabic Online text line recognition | 117 |
| 5.4.1 | Graphemes' Recognition (Fuzzy Classification) | 118 |
| 5.4.2 | Character Recognition (Graph-based Classification) | 119 |
| 5.4.2.1 | Character Grapheme-based Probability | 120 |
| 5.4.2.2 | Graph Construction | 120 |
| 5.4.2.3 | Path evaluation | 122 |
| 5.5 | Experimental Work | 125 |
| 5.5.1 | Grapheme's fuzzy models Generation | 125 |

| | | |
|--|--|-----|
| 5.5.2 | Performance evaluation..... | 126 |
| 5.5.3 | Experimentation using Arabic Online words | 127 |
| 5.5.4 | Experimentation using <i>Online-KHATT</i> database | 128 |
| 5.6 | Conclusions..... | 130 |
| CHAPTER 6 CONCLUSIONS AND FUTUER RESEARCH DIRECTIONS | | 132 |
| 6.1 | Conclusions..... | 132 |
| 6.2 | Limitations of the Work..... | 135 |
| 6.3 | Future Research Directions..... | 136 |
| REFERENCES | | 138 |
| VITAE..... | | 147 |

LIST OF TABLES

| | |
|--|-----|
| Table 2-1 The Arabic alphabet (الأبجدية العربية) | 10 |
| Table 2-2 Some secondary's used in Arabic handwritten text. | 11 |
| Table 2-3 Some of the Databases used in online Arabic recognition | 18 |
| Table 2-4 Summary of some techniques used for feature extraction for Arabic online text recognition | 33 |
| Table 2-5 Summary of some classification techniques for Arabic online text recognition | 41 |
| Table 3-1 Confusion matrix of the enhanced fuzzy classification..... | 61 |
| Table 3-2 The overall classification results of the human subjective evaluation | 62 |
| Table 3-3 Some misclassified samples of the fuzzy classifier..... | 64 |
| Table 4-1 The results of evaluating the generated classes | 80 |
| Table 4-2 The reduction of the codebook using bag-of-classes | 81 |
| Table 4-3 The impact of codebook size on recognition rate..... | 82 |
| Table 4-4 The Recognition Rates of Online Arabic Characters based on Graphemes Modeling | 88 |
| Table 4-5 Recognition rates with the datasets used in systems proposed for isolated Arabic online characters | 89 |
| Table 5-1 Characters' classes based on similarity in the main stroke | 95 |
| Table 5-2 Relative Heights of Arabic characters | 116 |
| Table 5-3 Results of Online Words Recognition Using HResult tool | 127 |
| Table 5-4 Results of Online Text Lines Recognition Using HResult tool | 128 |
| Table 5-5 Results of the more readable lines Using HResult tool | 129 |

LIST OF FIGURES

| | |
|--|----|
| Figure 2-1 Different online forms for Arabic character “Ain ع”. (a) Isolated (b) Ending (c) Middle (d) Beginning form. | 9 |
| Figure 2-2 Some Characteristics of Arabic handwritten text. It is cursive and written from right to left. (1) Diacritics (Dhamma, Shadda and fatha, left to right). (2) Dots (Triple, single and double, left to right) (3) Two ligatures. (4) The writing line. (5) Different Arabic letters with loops. | 12 |
| Figure 2-3 A general model of Arabic online text recognition system..... | 13 |
| Figure 2-4 A dataset entry for the ADAB database [10] | 16 |
| Figure 2-5 A sentence from OHASD database [12] | 16 |
| Figure 2-6 Example of smoothing the input data..... | 19 |
| Figure 2-7 Smoothing and Resampling Effect [18]..... | 20 |
| Figure 2-8 Illustrations of some of the pre-processing techniques [22] | 21 |
| Figure 2-9 Baseline detection for the Arabic word “مركز الشحيحة”[18]..... | 22 |
| Figure 2-10 Baseline estimation (a) Horizontal projection (b) baseline estimation for the three parts (c) final base line [30] | 23 |
| Figure 2-11 Arabic word Segmentation (a) Vertical segmentation. (b) Extra segmentation [31]..... | 24 |
| Figure 2-12 The Key Points of word لبيه [33] | 25 |
| Figure 2-13 Illustrations of some structural features. | 27 |
| Figure 3-1 Arabic (Indian) digits 0 to 9 | 44 |
| Figure 3-2 Overall block diagram of our approach for Arabic online digit recognition .. | 46 |
| Figure 3-3 Digit four (a) before simplification (b) after simplification..... | 47 |
| Figure 3-4 Digit six (a) before preprocessing and (b) after Preprocessing..... | 48 |
| Figure 3-5 Different samples of Arabic digit zero ‘0’ | 49 |
| Figure 3-6 (a) Standard writing direction with 22.5° gap (b) The overlap between direction 337.5° and direction 0° | 50 |
| Figure 3-7 (a) Sample of digit six ‘6’ with the directions of its segments (b) Segments histogram..... | 51 |
| Figure 3-8 Directional representations of several samples of digit three | 52 |
| Figure 3-9 MTL and Model Envelop for digit three..... | 53 |
| Figure 3-10 Generated Fuzzy models for digit ‘Three’ | 54 |
| Figure 3-11 Generated Fuzzy models for digit ‘Six’ | 55 |
| Figure 3-12 Fuzzy comparison between a sample and a model | 57 |
| Figure 3-13 Fuzzy classification block diagram | 59 |
| Figure 3-14 (a) Regular order of writing digit ‘three’ (b) Irregular order of writing digit ‘three’ | 63 |
| Figure 4-1 Overall block diagram of the proposed approach for Arabic online character recognition | 68 |
| Figure 4-2 InkMIPad Editor..... | 69 |
| Figure 4-3 Samples from the collected dataset. (a) Samples of different characters (b) Different samples from the end form of Arabic letter “Ain” | 70 |
| Figure 4-4 Preprocessing for the middle form of Arabic character ‘Ain’. (a) Before smoothing. (b) After smoothing..... | 71 |

| | |
|--|-----|
| Figure 4-5 Grapheme's Extraction. (a) The end form of character 'Ain'. (b) The extracted graphemes in different colors..... | 72 |
| Figure 4-6 partitioning the grapheme's trajectory into four windows..... | 73 |
| Figure 4-7 Samples of different graphemes (red curves) with lines connecting the grapheme's points with the center of gravity (blue lines)..... | 74 |
| Figure 4-8 Freeman chain code..... | 75 |
| Figure 4-9 Curliness feature..... | 76 |
| Figure 4-10 Graphemes' Codebook..... | 78 |
| Figure 4-11 Grapheme' location | 85 |
| Figure 4-12 Sample of grapheme (the blue curve) where S represents the start-point, E represents the end-point, M_1 is middle-point of the line connecting the start-point with the end-point and M_2 is middle point of the grapheme..... | 86 |
| Figure 5-1 General framework for the proposed Arabic online text recognition | 93 |
| Figure 5-2 Arabic Online text from Online KHATT database with some segmented characters | 94 |
| Figure 5-3 Letter FAA (a) before the simplification (b) after the simplification | 96 |
| Figure 5-4 Some Arabic Online characters with their segmented graphemes..... | 96 |
| Figure 5-5 Curvature of Points P1, P2 and P3 | 97 |
| Figure 5-6 The sharp turns of Arabic PAW "صا" (marked by small red circles)..... | 97 |
| Figure 5-7 Angle Quantization feature | 98 |
| Figure 5-8 The difference between the curvature (\emptyset) and Writing direction (Θ)[88].... | 100 |
| Figure 5-9 Grapheme's End-Points Feature | 101 |
| Figure 5-10 Graphemes' Codebook..... | 102 |
| Figure 5-11 Reduced Codebook (Bag-of-Classes) | 102 |
| Figure 5-12 Codebook generated from the selected features..... | 104 |
| Figure 5-13 Arabic Online text line (a) before preprocessing (b) after preprocessing.. | 106 |
| Figure 5-14 Baseline Estimation..... | 107 |
| Figure 5-15 The delayed stroke of letter TAA has grater size than the main stroke of letter ALEF..... | 107 |
| Figure 5-16 Text line with windows drawn around its strokes..... | 108 |
| Figure 5-17 (a) The height of the delayed stroke of letter ALEF is greater than the height of letter WAW. (b) Partial overlap between the main strokes of letters WAW and MEEM. | 108 |
| Figure 5-18 Different Arabic online characters with loops | 111 |
| Figure 5-19 The text line after finding the loops | 111 |
| Figure 5-20 Closing Open-loop | 112 |
| Figure 5-21 Text line after closing the loop of letter TAA..... | 113 |
| Figure 5-22 Possible segmentation points | 113 |
| Figure 5-23 Estimating the height of the Text Line (a) Height's Estimation based on using one window (b) Estimation based on using a window for each stroke and taking the height of the maximum one | 115 |
| Figure 5-24 Illustration of steps to recognize Arabic Online line from Online KHATT database..... | 117 |
| Figure 5-25 Trapezoidal membership function..... | 118 |
| Figure 5-26 Example to show the steps of mapping the graphemes to their corresponding characters | 124 |

| | |
|---|-----|
| Figure 5-27 Initial steps of constructing a graph for recognizing a text line | 125 |
| Figure 5-28 Some samples from the collected Arabic Online words. | 127 |
| Figure 5-29 Loop is drawn with character that do not require loop | 130 |

ABSTRACT

Full Name : Dhiaa Abdulrab Ali Musleh
Thesis Title : Online Arabic Handwritten Text Recognition using Syntactical–Based Techniques
Major Field : Computer Science and Engineering
Date of Degree : May, 2016

In recent years tablets and smartphones have been used widely. Online text forms a natural representation for inputting data to these devices. Developing a system for automatic recognition of online text provides a quick and natural way of communication between these devices and human beings. Statistical-based approaches have been widely used in research on online text recognition while syntactical-based approaches have remained less explored. In this thesis, research on automatic Arabic online text recognition using syntactical-based techniques has been conducted.

An improved fuzzy modeling approach of Arabic text is proposed. This approach is applied to Arabic (Indian) online digits' recognition. In this approach, fuzzy models for the different digits are automatically generated using the training data. The fuzzy intervals are generated automatically based on the analysis of the training samples at the digit segment level. In addition, we automatically generate weights for the different segments using the training samples. These weights are integrated to the fuzzy similarity estimate. These fuzzy models proved to be able to handle the variability of the handwriting styles.

A grapheme-based approach for recognizing isolated online Arabic characters is presented. The proposed approach models the online characters based on their graphemes using generated codebook. Different features and classification approaches are used in order to investigate the proposed modeling on Arabic online character recognition.

A new algorithm for segmenting the Arabic online text into its graphemes is proposed. The algorithm utilizes the way characters are joined in Arabic online text. Based on this algorithm, a grapheme-based approach for Arabic online text recognition is presented. A fuzzy classification approach is used to recognize the extracted graphemes from the testing data. A graph-based approach is used to map the recognized graphemes to their corresponding characters based on the graphemes' statistics gathered by analyzing the training data.

The proposed techniques in this thesis are applied on Arabic online digits, characters, words and text databases and the obtained results are promising. The presented work is easily extendable.

ملخص الرسالة

الاسم الكامل: ضياء عبدالرب علي مصلح

عنوان الرسالة: التعرف الآلي على الكتابة العربية اليدوية الآنية باستخدام الطرق البنوية

التخصص: علوم وهندسة الحاسب الآلي

تاريخ الدرجة العلمية: مايو 2016

أصبحت الأجهزة المدعومة بالأقلام الإلكترونية ، مثل الأجهزة اللوحية و الهواتف الذكية، تستخدم بشكل واسع مؤخراً. إن استخدام الكتابة الآنية بالقلم الإلكتروني يمثل الطريقة الأنسب لإدخال البيانات في هذه الأجهزة. سيوفر تطوير نظم للتعرف الآلي على النصوص المكتوبة باستخدام القلم الإلكتروني طريقة سريعة وسهلة لتعامل المستخدمين مع هذه الأجهزة. استخدمت الطرق الإحصائية في الأبحاث المتعلقة للتعرف على النصوص الآنية المكتوبة باستخدام الأقلام الإلكترونية بشكل واسع، ولكن الطرق التي تعتمد على الصفات البنوية لم تستخدم بشكل كاف. قمنا في هذه الرسالة بدراسة التعرف الآلي على النصوص العربية الآنية المكتوبة باستخدام الأقلام الإلكترونية باستخدام الصفات البنوية.

تم في هذه الرسالة إقتراح طريقة مطورة لاستخدام التمثيل الضبابي (Fuzzy modeling) في التعرف على الكتابة، وقد تم تطبيق هذه الطريقة لتمثيل الأرقام العربية (الهندية). تقوم هذه الطريقة بتوليد نماذج للأرقام العربية بشكل آلي بالاستفادة من البيانات المستخدمة في مرحلة التدريب. بالإضافة إلى ذلك، يتم إعطاء أوزان (Weights) مختلفة لأجزاء الرقم حسب أهميتها بحيث يستفاد من هذه الأوزان في مرحلة تقييم التشابه. إن النماذج المقترحة قادرة على التعامل مع التغيرات الناتجة من أنماط الكتابة اليدوية المختلفة. تعرض هذه الرسالة أيضاً طريقة للتعرف على الحروف العربية المنفصلة اعتماداً على تمثيل الحروف باستخدام مكوناتها الأساسية (Graphemes). وقد تم تقييم الطريقة المقترحة باستخدام مجموعة من السمات بالإضافة إلى مجموعة من تقنيات التصنيف الحديثة.

تقدم هذه الرسالة أيضاً خوارزمية لتقسيم النصوص العربية الآنية المتصلة إلى أجزاءها الأساسية. تم تصميم هذه الخوارزمية بحيث تراعي طريقة ربط الحروف ببعضها في الكتابة العربية. تم الاستفادة من الخوارزمية المقترحة في تصميم تقنية للتعرف الآلي على النصوص العربية الآنية باستخدام سماتها البنوية. تعتمد التقنية على التصنيف الضبابي

(Fuzzy classification) في التعرف على الأجزاء الأساسية التي يتم تحديدها من النصوص ومن ثم يتم التعرف على الحروف المقابلة لهذه الأجزاء بالاستفادة من الإحصائيات الناتجة من تحليل الحروف المستخدمة في مرحلة التدريب.

تم تطبيق التقنيات المقترحة في هذه الرسالة على قواعد بيانات تشمل الأرقام والحروف والكلمات والنصوص العربية الأنوية، وقد حصلنا على نتائج واعدة. بالإضافة إلى ذلك فإن التقنيات المقترحة قابلة للتطوير والذي بدوره سيؤدي إلى تحسين النتائج.

CHAPTER 1

INTRODUCTION

Nowadays, the use of smart phones and tablets is growing rapidly. Handwriting is commonly used in these devices for recording information. As a result, online handwriting recognition gained a significant interest in recent years. In reality, research on online recognition has started during the 1960s [1] and is still gaining powerful attention among scientist due to the development of new more tablets and smart phones and the wide use of these devices. The pen movement, which represents the text, is captured in these devices as a sequence of x and y coordinates. Automatic recognition of handwritten Arabic characters is used in a variety of applications such as forms filling, text editing and note taking. A reliable online recognition system with acceptable recognition rate is needed as an alternative input method.

Online Arabic handwritten recognition has been less researched compared to offline Arabic handwritten recognition. The reasons for this may be attributed to the challenges related to online Arabic handwriting recognition systems. Some of these challenges include the need for special hardware; techniques of other languages may not work for online Arabic text recognition; etc.

In this thesis, we conduct research on Arabic online text recognition. We apply structural-based techniques to Arabic online text recognition since they proved satisfactory with offline Arabic handwritten text.

1.1 Handwriting Recognition

Handwriting recognition is a process of translating a text written in its spatial or temporal form into its symbolic representation [2]. Handwriting recognition is a hot research direction in pattern recognition and has become a topic of intensive study due to its important applications such as forms processing, automatic cheques processing, postal address recognition, writer identification etc.

1.2 Offline vs. Online Handwriting Recognition

The field of handwriting recognition can be classified into two different types based on the nature of the handwritten text: *off-line* recognition and *on-line* recognition. Off-line recognition addresses the recognition of handwriting in the form of scanned images whereas on-line recognition deals with the recognition of handwriting in the form of (x, y) coordinates. The handwritten text is collected, in offline recognition, using an ordinary pen and paper before it's converted into an image by using a scanner or a camera. Whereas special devices, such pen-based or touch-screen computers/smartphones, are need in online recognition to capture the x and y coordinates of the text as a function of time. In contrast to offline recognition, online recognition captures the dynamic information of the written text such as the speed and the direction of the writing [3]. However online text dose not clearly represent some spatial information that is available in offline text. For example, if a letter, like BAA “ب”, is written then different strokes are written before writing the ‘dot’

of that letter, here the letter will be far from its dot in the time domain although they are close to each other in the spatial domain [4].

1.3 Challenges in Arabic online handwriting recognition

In the literature, most of the effort in Arabic text recognition has been spent on off-line Arabic printed and handwritten text; whereas comparatively fewer studies focused on Arabic online handwritten text. The reasons for this may be due to the challenges related to online Arabic handwriting recognition systems. The main challenges of online Arabic handwriting recognition are as follows:

1. Online Arabic text recognition needs special hardware.
2. The writing of online devices is less controlled than writing using a pen on paper. Consequently, the variability between writers will be increased and even with the same writer. This makes the problem of online handwriting recognition a challenging pattern recognition problem. However, online handwritten has the time of the writing information (sequence of writing) available.
3. The available classifiers of other languages may be used for off-line text recognition while structural classifiers are needed for Arabic online text recognition. The structural approaches are language dependent. So, techniques of other languages may not suit Arabic text recognition.
4. There is no comprehensive and freely available database for online Arabic text recognition research.

1.4 Motivation

Arabic Language is spoken by more than 422 million all over the world and is considered as the official language of the Arab world [5]. Moreover, Arabic characters are used in several languages besides Arabic such as Urdu, Persian, Kurdish and Malay. Therefore, automatic processing of Arabic text has widespread benefits. Computers are greatly affecting our way of life and their usage is increasing rapidly. It is very important for simplifying the way of exchanging information between users and computers since input devices today, such as keyboards, have some limitations. Nowadays, pen-based computers (tablet PCs) are used widely. Personal digital assistants and smart phones with pen or touch screen have become popular as input devices. The presence of online handwritten text recognition systems is very important to provide a natural and convenient way of two communications between users and computers. Much researches has conducted on Arabic handwritten recognition. Most of this research addressed Offline Arabic handwritten text. While online Arabic handwritten recognition is less researched. Developing a system for automatic recognition of online Arabic text will provide a quick and natural way of communication between computers, digital assistants, smart phones and human beings.

1.5 Contributions of the Thesis

In this thesis, we conducted research on automatic recognition of Arabic online text. The main contributions of this work are summarized in the following paragraphs

1. A literature review of Arabic Online text recognition.

A literature review is conducted to survey the published work in the field of Arabic online text recognition. It includes a review for the different phases of Arabic online text recognition systems and also a study for the different attempts to build databases for Arabic online text. Details of this literature review are presented in chapter 2.

2. A novel fuzzy modeling approach applied to Arabic (Indian) online digits' recognition:

a) Robust fuzzy models for Arabic online digits.

We automatically generate robust fuzzy models for Arabic online digits using the training data. The fuzzy durations are automatically generated and are set at the digit segment level. In addition, automatic generation of weights at the segment level, which indicate the importance of the different segments of the digit for recognition, was integrated into the models. These fuzzy models are considered as an improvement over previous works as discussed in chapter 3.

b) A robust two cascaded stages digit classification.

The proposed classification phase consists of two cascaded stages, where in the first stage the system classifies digits into zero/nonzero classes and the second stage classifies digits 1 to 9.

c) A novel fuzzy syntactic classifier with integrated feedback stage.

We develop a novel fuzzy syntactic classifier with integrated feedback stage that verifies the selected classes for test samples using segment histogram features. This stage improved the syntactic fuzzy classification.

The Experimental evaluations of the proposed approach show very promising results for Arabic online digits.

3. A new grapheme-based modeling approach applied to Arabic online character recognition:

a) A new approach for modeling Arabic online character using graphemes.

We propose a new approach for modeling the Arabic online characters based on their graphemes using the generated grapheme codebook.

b) A new grapheme-based classification approach for Arabic online characters.

We propose a new classification approach for Arabic online characters based on grapheme modeling. The graphemes of the testing character are extracted and then used to build the pattern of the corresponding character. This is not a trivial process since deciding which grapheme belongs to which character is a challenging task and it needs an effective algorithm [6].

4. A new approach for Arabic online text recognition using grapheme-based modeling and fuzzy classification:

a) A new approach for segmenting the Arabic online text into its graphemes.

A new approach for segmenting the Arabic online text into graphemes is proposed. The proposed approach is designed to consider the way characters are joined in Arabic online text.

b) A new approach for modeling Arabic online text.

We propose an approach for modelling the Arabic online text based on its graphemes. The grapheme models are built, in the training phase, from pre-segmented Arabic online characters while Arabic online text lines are used in the testing phase. Therefore, the training dataset is different from the set used for testing. The grapheme models are built in a fuzzy manner to better address the problem of variability in writing.

c) A similarity/dissimilarity measure for recognizing the graphemes extracted from the testing lines.

This similarity/dissimilarity measure is based on our proposed fuzzy classification. The weight of the similarity between a testing grapheme and a model is represented by the membership value assigned to that grapheme. The calculations of membership values are based on automatically generated durations and weights, and not fixed.

d) A graph-based classification for recognizing text characters.

Our approach for recognizing the characters is based on construction of a graph where nodes represent the candidate recognized characters and edges are created between nodes depending on the grapheme-based probabilities calculated from the training data. Each path in the graph represents a possible solution for the input text. The probabilities assigned to the path's edges are used to find the best path that represents the recognized text.

1.6 Organization of this Thesis

The rest of this thesis is organized as follows. In Chapter 2 we discuss the characteristics of Arabic language script, then we present a literature review of the published work in the field of Arabic online text recognition. Chapter 3 describes the proposed fuzzy technique for Arabic (Indian) online digits recognition. The proposed automatic fuzzy model generation approach is described in detail. In addition, this chapter presents the proposed fuzzy classification approach. A grapheme-based approach for recognizing isolated online Arabic characters is presented in Chapter 4. Chapter 5 presents the proposed Arabic online text recognition. A new approach for segmenting the Arabic online text into its graphemes is presented. In addition, the details of the training and testing phases are described in detail. Chapter 6 concludes this thesis by addressing the outcomes and future work.

CHAPTER 2

LITERATURE REVIEW

2.1 CHARACTERISTICS OF ARABIC SCRIPT

The Arabic Language is one of the most widely spoken and studied languages in the world. The Arabic script has many characteristics which make it distinguished from other languages. The Arabic alphabet consists of 28 basic characters. Arabic text is written from right to left and it is cursive in both handwritten and printed text. Each Arabic character has at least two and at most four different forms based on its position in the word. Figure 2-1 shows the four different forms of online Arabic “Ain ع” character. These forms are: isolated, ending, beginning and middle form.



Figure 2-1 Different online forms for Arabic character “Ain ع”. (a) Isolated (b) Ending (c) Middle (d) Beginning form.

Some Arabic characters have one, two or three dots. These dots help in differentiating between the characters that are sharing the same basic shape as in BA’A (ب), TA’A (ت) and THA’A (ث). Table 2-1 shows the Arabic alphabet (الأبجدية العربية).

Table 2-1 The Arabic alphabet (الأبجدية العربية)

| Transcription | Printed form | Online (Handwritten) form | Transcription | Printed form | Online (Handwritten) form |
|---------------|--------------|---------------------------|---------------|--------------|---------------------------|
| ALEF | أ | ا | DHAD | ض | ظ |
| BAA | ب | ب | TTAA | ط | ط |
| TAA | ت | ت | TTHAA | ظ | ظ |
| THA | ث | ث | AIN | ع | ع |
| JEEM | ج | ج | GHAIN | غ | غ |
| HAA | ح | ح | FAA | ف | ف |
| KHAA | خ | خ | QAAF | ق | ق |
| DAL | د | د | KAAF | ك | ك |
| THAL | ذ | ذ | LAM | ل | ل |
| RAA | ر | ر | MEEM | م | م |
| ZAIN | ز | ز | NOON | ن | ن |
| SEEN | س | س | HHAA | هـ | هـ |
| SHEEN | ش | ش | WAW | و | و |
| SAAD | ص | ص | YAA | ي | ي |

Some Arabic letters (e.g. ك, و, أ) may have a zig-zag like stroke called (Hamza ء). The Hamza's and dots are called secondary's and are written either above the main stroke like in WAW (و) and Noon (ن), or in the middle as in JEEM (ج) and KAAF (ك) or below the

main stroke like ALEF (ا) and BAA(ب). Some characters are connected to the previous and/or the next characters within the same word, and some are not connected.

Arabic Characters in a word may have extra strokes called diacritics such as Fat-hah (َ), Kasrah (ِ) and Dhammah (ُ). Tanween which is written as double Fat-hah (ً), Dhammah (ٌ), or Kasrah (ٍ) is considered as another form of diacritics. These diacritics will not only change the pronunciation of the word but also can change the meaning such as the word (كتب) can be pronounced as either (كَتَبَ kataba) which means ‘he wrote’ or (كُتِبَ kutiba) which means ‘it was written’ or (كُتُب kutub) which means ‘books’. Although these diacritics are important, the Arabic readers are familiar with reading a text without diacritics and understanding the meaning from the word’s context. Table 2-2 illustrates the different shapes of the possible secondary’s used in Arabic writing.

Table 2-2 Some secondary’s used in Arabic handwritten text.

| Secondary’s | Shape | Position | Examples |
|----------------------|-------|--------------------|----------|
| Single dot | . | Above/middle/below | ز ج ب |
| Double dots | ـ ـ | Above/below | ي ت |
| Triple dots | ـ ـ ـ | Above | س ش |
| Shadda | ّ | Above | طّ |
| Hamza | ء | Above/below | أ إ |
| Fatha / Double Fatha | / / | Above | بَ بً |
| Kasra / Double Kasra | / / | Below | بِ بٍ |
| Dhamma/Double Dhamma | / / | Above | بُ بٌ |
| Madda | ~ | Above | آ |

In Arabic text, there is also the ligature which is formed by overlapping two or more letters, such as laam–alif “لا”, meem-haa in “مها” and so on.

There are some advantageous features in the Arabic scripts that may help in Arabic handwriting recognition. One of these features is the existence of the handwriting baseline which is a virtual horizontal line on which the letters are connected to each other. In addition, there are some rules that apply for connecting the letters in both printed and handwritten Arabic text. Figure 2-2 illustrates some characteristics of Arabic script.

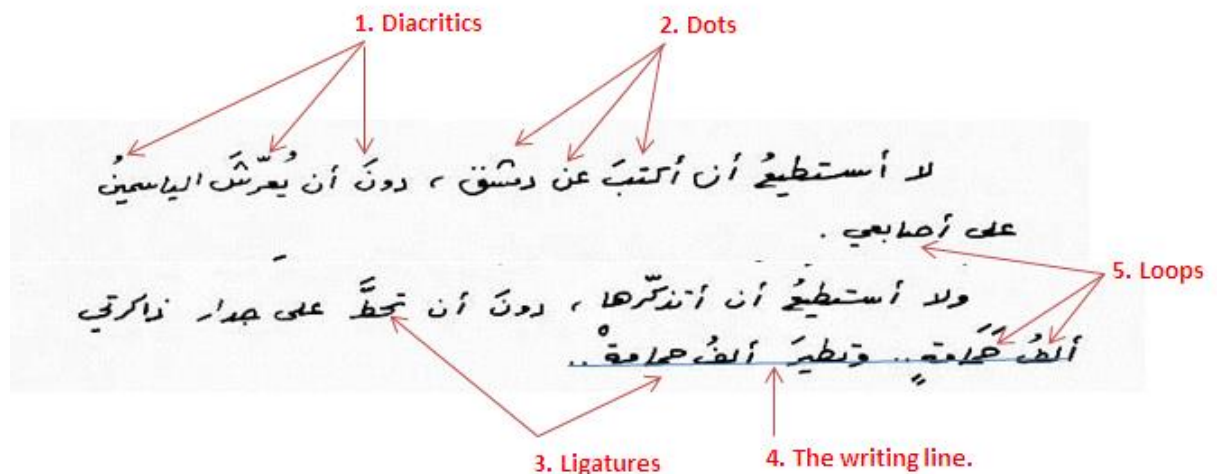


Figure 2-2 Some Characteristics of Arabic handwritten text. It is cursive and written from right to left. (1) Diacritics (Dhamma, Shadda and fatha, left to right). (2) Dots (Triple, single and double, left to right) (3) Two ligatures. (4) The writing line. (5) Different Arabic letters with loops.

In general, these characteristics of Arabic text (like the cursive nature of the language, the similarity of groups of symbols and the overlapping between characters) make the automatic recognition of Arabic text more difficult comparing to the automatic recognition of Latin text. For more details, reference may be made to [7] and [8].

2.2 General model for Arabic online text recognition system

A general model for an online Arabic handwriting recognition system is presented in figure 2-3. The presented system is composed of several phases; however these phases are not necessarily present in all systems. The input for the system is the (x, y) coordinates of the points of the writing trajectory as a function of time. This input may need to be enhanced by applying preprocessing steps such as remove duplication of points, smoothing, size normalization and re-sampling, etc. Then, the enhanced data may need to be segmented into smaller parts (such as lines, words or characters). After that, the features are extracted from the segmented data. These features are used to train the system and build the data models. In the classification phase, the extracted features are used to classify the testing data based on the generated models. The last phase, post-processing, enhances the recognition rate by refining the results of the classification stage.

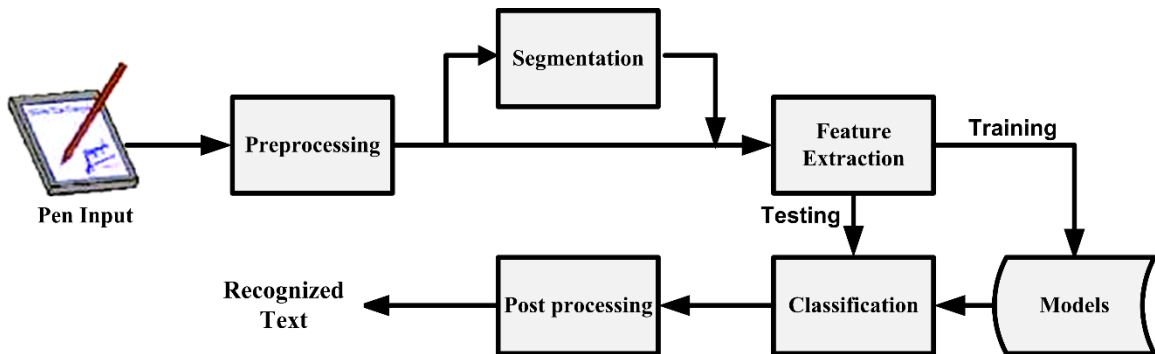


Figure 2-3 A general model of Arabic online text recognition system

The accuracy of each phase has an important impact on the overall recognition performance. In addition, each phase has its own difficulties and challenges that need to be addressed in order to improve the overall accuracy. Different approaches have been

proposed for each phase in the literature. The different phases and their proposed approaches will be discussed in more details in the next sections.

2.3 Arabic online handwritten databases

Attempts to develop efficient Arabic online handwriting recognition systems raise the need to collect and build comprehensive online Arabic databases. These databases are very important for researchers in this area. Using one benchmark database, the researchers can evaluate their own systems in relation to the state of the art.

Systems that are evaluated based on private, self-generated, databases are not comparable; this is because in some systems, better results may be obtained due to high quality of the database used to evaluate the system. Therefore, common benchmarking databases are essential for the research community to perform comparative evaluation. This section discusses the available online Arabic text databases that are used by the research community.

We are not aware of any standard comprehensive database for online Arabic text recognition that is available freely for researchers. The published work on Online Arabic handwritten text is mostly based on self-generated database, this is evident from Table 2-5 which shows various online databases used by researchers in their experimentations. However, some attempts to build benchmarking databases for Arabic online handwritten text have been realized in recent years.

A real attempt to build comprehensive databases was in 2008 by Kherallah et al. [9]. They developed the on/off (LMCA) dual Arabic handwriting database in REGIM (REsearch

Group on Intelligent Machines) laboratory. LMCA is the abbreviation of a French sentence "Lettres, Mots et Chiffres Arabe" which is composed of letters, words and digits. This database consists of (100,000 letters), (500 words) and (30,000 digits) written by 55 writers. Since this database is developed for both online and offline handwritten recognition, two procedures were used to collect the data. In the first procedure, text's trajectory is collected by taking the (x, y) coordinates for online recognition. For offline recognition, the second procedure collects the images of the handwritten trajectory. This database does not represent a natural Arabic online text lines as only words, letters and digits are included.

The ADAB (Arabic DAtaBase) was developed in 2009 by a collaboration between the Institute for Communications Technology (IfN) and the Ecole Nationale d'Ing'nieurs de Sfax (ENIS) [10]. The ADAB in version 1.0 consists of 15,158 Arabic words of 937 Tunisian city names collected from more than 130 writers. Figure 2-4 illustrates the dataset entry for each written city name in the ADAB database. This database is not a natural online Arabic text database as it consists of city names only.

Due to the lack of a comprehensive real database, Saabni and El-Sana propose a first synthetic database for online Arabic handwritten words [11]. The words were generated by using a set of Arabic words written by 10 writers and including all the Arabic characters in their forms. Based on these words, their system generates the shapes of all words in the database, for each writer. The generated synthetic database contains 300,000 words consisting of 48,000 word-parts (PAWs). This database has two limitations, it is synthesized and it consists of words and not natural Arabic online text.


| | |
|-------|---|
| Word | أعطاية |
| Image |  |
| Label | العطاية |
| inkML | <pre> <trace duration="93" id="id001" type="penDown"> 531 44,531 42,531 41,532 40,532 42,533 47,534 54,535 62,536 65,536 68,537 70,537 72,537 74,537 75,537 76,538 76</trace> ... <trace duration="15" id="id013" type="penDown">350 96,350 97</trace> <trace duration="94" id="id015" type="penDown">350 96,342 99,341 99,340 100,340 101,340 102,339 102</trace> <trace duration="0" id="id017" type="penDown">303 56,303 58</trace> <trace duration="93" id="id019" type="penDown">303 56,294 59,293 60,292 61,292 62,291 63,290 63</trace> </pre> |

Figure 2-4 A dataset entry for the ADAB database [10]

In 2010, an Online Arabic Handwritten Sentence Database (OHASD) was developed by Elanwar et al. [12]. Sentences of this database were sampled from daily newspapers. The OHASD comprises 154 paragraphs which consist of about 3,800 words and about 19,400 characters written by 48 writers. The writers are well-educated and they are from different research centers of ages between 23 and 40 years of both genders. Figure 2-5 shows a sentence from OHASD Database.

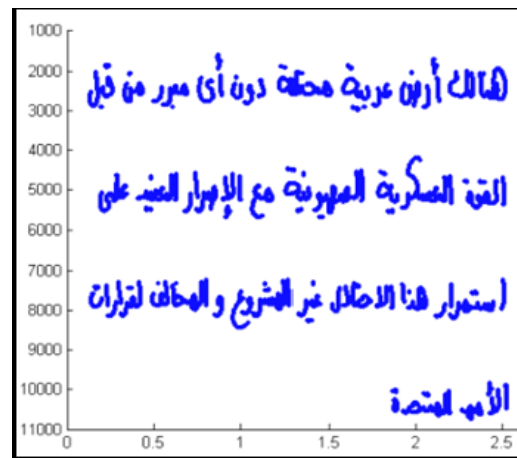


Figure 2-5 A sentence from OHASD database [12]

Another attempt to create a comprehensive database for online handwritten Arabic language was proposed by Arabic Language TEchnology Center (ALTEC) in 2011 [13]. The proposed ALTEC database includes 5,000 pages of online Arabic documents written by 1000 different writers. The whole database consists about 35,000 lines; each line represents a sentence. It includes 175,000 words consisting of 500,000 PAWs and about one million letters.

Another attempt to create a larger database for Arabic On-line handwritten Digits was made in 2012 by Azeem et al. [14]. They proposed an Arabic On-line Digits Database (AOD). AOD was collected from 300 different writers varying over different ages and without enforcing any constraints on the digit size, orientation or number of strokes per digit. More than 30,000 online Arabic digits were collected by asking each writer to write an average of 10 samples per digit. The AOD is freely available for noncommercial research at: <http://www.aucegypt.edu/sse/eeng/Pages/AOD.aspx>. Table 2-3 summarizes some of the used databases in online Arabic recognition.

2.4 Preprocessing techniques

Preprocessing is a basic step in any handwriting recognition system in which the acquired data is enhanced to achieve better recognition rates. Basically, the data can be enhanced by smoothing, reducing the noise, normalizing the trace, etc.

Table 2-3 Some of the Databases used in online Arabic recognition

| Database | Description | Writers | Scope | Comment |
|---------------|--|---------|---------------------------|-----------------------|
| LMCA | 100,000 letters 500 words 30,000 digits | 55 | letters, words and digits | Online and Offline |
| ADAB | 15,158 Arabic words of 937 Tunisian town names | 130 | Words | ----- |
| Saabni et al. | 300,000 words containing 48,000 word-parts | 10 | Words | Synthetic |
| OHASD | Newspaper's sentences, 154 paragraphs, 3800 words and more than 19,400 characters | 48 | Sentences | ----- |
| ALTEC | 5,000 pages, 35,000 sentences, 175,000 words that include about 500,000 PAWs and about 1 million letters. | 1000 | Sentences | ----- |
| AOD | 30,000 online Arabic digits | 300 | Digits | ----- |

In online text recognition, each digit/character/word is represented by a sequence of points (i.e. x and y coordinates). These points can contain considerable amount of noise which makes the task of handwriting recognition more challenging. This noise may be caused by the online devices as well as by a hand jerk. Several preprocessing steps can be applied on the online writing such as smoothing, re-sampling to eliminate the noise and size normalization. Figure 2-6 shows an example of input Arabic letter before and after preprocessing phase.

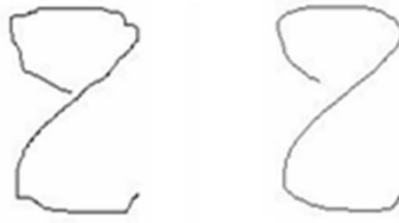


Figure 2-6 Example of smoothing the input data

In the following subsections we discuss the research related to the preprocessing phase of Arabic online text. We classify these tasks into two categories; resampling and noise removal category and baseline detection category.

2.4.1 Resampling and noise removal

In general, the captured (x, y) coordinates are not divided evenly along the online text trajectory because of the variation in writing speed. Therefore, the number of acquired points varies, depending on writing speed (usually there are more points where the speed is low and fewer points where the speed is high). Resampling is performed to remove the variability due to writer's speed. Mezghani et al. normalized the size of the acquired string of coordinates by replacing the acquired points with a sequence of a fixed number of equidistant points, namely 30 points [15].

A resampling algorithm based on trace segmentation method was adopted by Kosmala et al. to overcome the variations in writing speed by resampling the trajectory points spatially with fixed length vectors [16]. The main disadvantage of this algorithm is deciding the distance between each two points, which is called the resampling distance. This distance

needs to be tuned either empirically or using some criteria to determine the distance [17]. This distance needs to be adjusted carefully. If it is large, that means the number of points is decreased which may cause some information to be lost. If the distance is small, the number of points is increased without increasing the relevant information. Pastor et al. try to overcome this problem by normalizing the writing speed instead of doing trace segmentation. In their approach, the derivatives are normalized by the derivative module at each point [17]. This normalization approach was adopted by Eraqi and Abdulazeem [18] to normalize the writing speed and enforce equal distance between the data points as shown in figure 2-7.

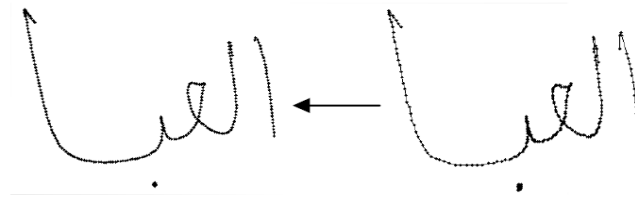


Figure 2-7 Smoothing and Resampling Effect [18]

Linear interpolation was used by Azeem et al. to solve the problem of large distances between consecutive points [14].

Tagougui et al. normalized the handwriting script size by first adjusting the vertical dimension of the script lines, then replacing each point's coordinates (x ,y) as follows; $Normalized_x = 128 * ((x - Minimum_x) / m)$; and $Normalized_y = 128 * ((y - Minimum_y) / m)$; Where m is the length of the maximum dimension and 128 is a threshold selected empirically [19].

Alsallakh and Safadi normalized the input stroked into a 100x100 pixel box to achieve size independence [20]. Al-taani instead used a bounding box of 5x5 drawn around the

character [21]. Similarly, Daifallah et al. scale every letter and center it in a window frame with the size 128×128 [22]. Point clustering and de-hooking were also used in this work. Point clustering is used to remove unnecessary points by averaging the neighboring points whereas de-hooking is performed to remove the part of the stroke that contains hooks. Figure 2-8 illustrates the results of these preprocessing techniques implemented in [22]. De-hooking were also used by Husain et al. [23].

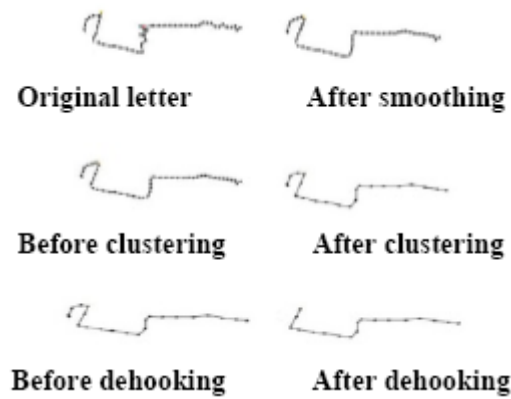


Figure 2-8 Illustrations of some of the pre-processing techniques [22]

Smoothing is an important factor for good segmentation and high recognition rate. In the literature, different techniques have been proposed for smoothing. Mezghani et al. smooth the online character signals to remove noise by averaging a point with its 3 neighbors [15]. The same approach is also applied by Eraqi and abulazeem by replacing each point with the mean value of itself and its neighbors [18].

Chebyshev filter is used by kherallah et al. to eliminate duplicated data points [24]. Chebyshev filter with a radius of filtering window equal to 8 is also used in [19] [25].

Teredesai et al. converted the online stroke information into an offline image. Then offline preprocessing techniques were applied to the converted image [26].

Horizontal and Vertical Projection Profile was used by Ismail and Abdullah to help in determining the character's shape and the position of its dots [27]. Laplacian Filter was also used in the preprocessing phase of their work.

2.4.2 Baseline Detection

Estimating the baseline of written words is considered as a common preprocessing step for online Arabic text. Basically, the baseline is the horizontal line on which the characters of cursive text are connected and aligned. An approach for estimating the baseline in both offline and online written words is presented by Boubake et al. in [28]. Their approach begins by grouping points according to aligned neighborhood. Then the estimated baseline is enhanced based on some topological conditions. Razzak et. al presented a baseline detection algorithm which is composed of three stages [29]. In the first stage, the secondary strokes are segmented from the raw data. Then, the primary baseline is estimated in the second stage using the horizontal projection on the ghost shape. Finally, the local baseline is detected based on the primary baseline and fetures extracted from ending shape of the word. The baseline was used by Eraqi and Abdelazeem in order to detect delayed strokes and construct Part of Arabic Word (PAW) stroke [18]. Baseline detection algorithm using horizontal projection histogram was used in their work as shown in figure 2-9.

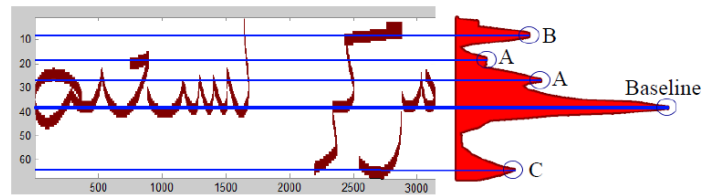


Figure 2-9 Baseline detection for the Arabic word “مركز الشيخية” [18]

In 2011, Abdel Azeem and Ahmed converted the online text trajectory into its corresponding bitmap image; then a horizontal projection algorithm was used for estimating the baseline as shown in Figure 2-10.a [30]. In order to improve the estimation of the baseline, the trajectory image was divided into three vertical parts and the baseline was estimated for each part as shown in Figure 2-10.b. Their experiments showed that the best estimation for the baseline is the minimum one among the four lines, i.e. Final-Baseline = min(baseline of whole image, baseline part1, baseline part2, baseline part3), as shown in Figure 2-10.c.

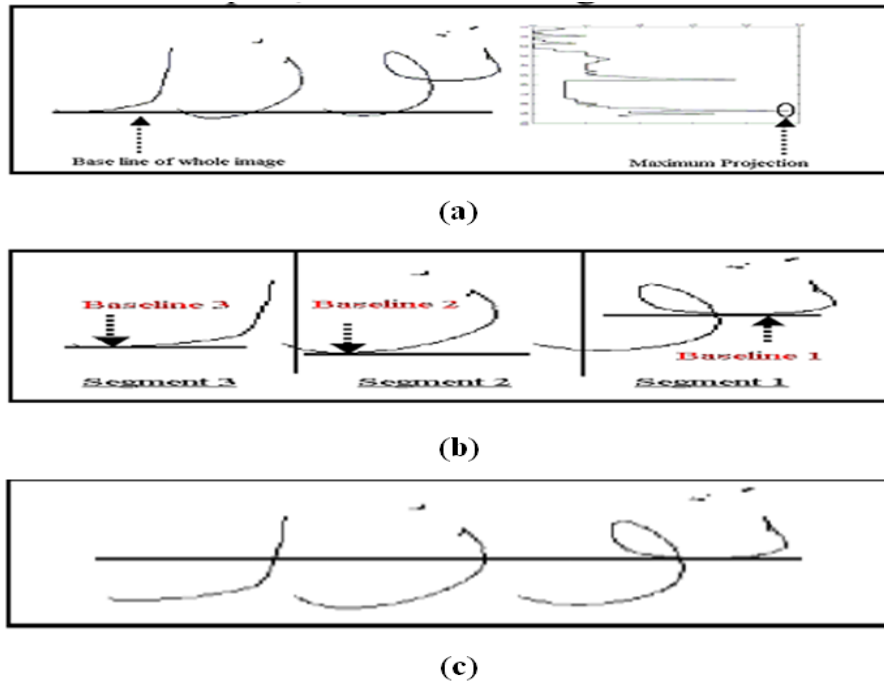


Figure 2-10 Baseline estimation (a) Horizontal projection (b) baseline estimation for the three parts (c) final base line [30]

2.5 Segmentation techniques

Segmentation of cursive words into characters or strokes may be needed to recognize online words. Daifallah et al. proposed their own segmentation algorithm called KHDJ-1 [22]. KHDJ-1 segments strokes into letters in four stages: arbitrary segmentation, then segmentation enhancement followed by consecutive joints connection and finally locating the segmentation point. This algorithm has the advantage of producing at least all correct segments. Sternby et al [31] segmented the input at the vertical extreme points with respect to the writing direction [31]. Then, a set of heuristic rules are applied to trigger additional segmentation points as shown in figure 2-11.

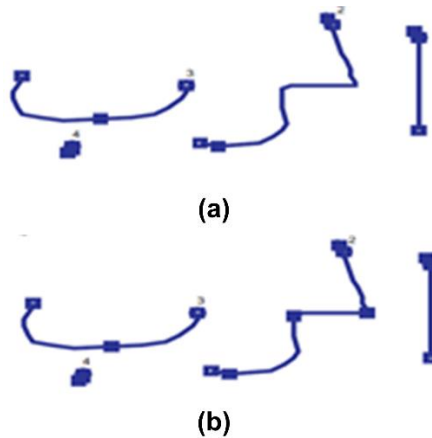


Figure 2-11 Arabic word Segmentation (a) Vertical segmentation. (b) Extra segmentation [31]

Boubaker et al. proposed fuzzy graphemes segmentation. They segment the words into graphemes based on extracting two types of points: the bottom of the valleys close to the writing line and the angular points [32].

In 2011, a segmentation algorithm based on segmenting each PAW stroke into its basic graphemes was proposed by Eraqi and Abdelazeem [18]. The thresholds of the proposed algorithm were defined based on statistics made on the ADAB database. The graphemes, generated using this algorithm, are not affected by baseline detection errors since this algorithm is independent of the baseline.

A real-time segmentation approach was proposed by George Kour and Raid Saabne in 2014 [33]. The proposed approach has three phases, in the first one the potential Segmentation Points SPs are nominated and scored while the stroke is being written. The second phase starts once the whole stroke is available and uses a set of rules to eliminate redundant SPs and rescore the strokes. The strokes scores are used in the final phase to yield the final set of SPs. Figure 2-12 shows the Key Points, KP, which are the possible segmentation points of word لبيه (colored in red). The green points in the figure (KP₀ and KP₅) represent the first and the last points in the stroke

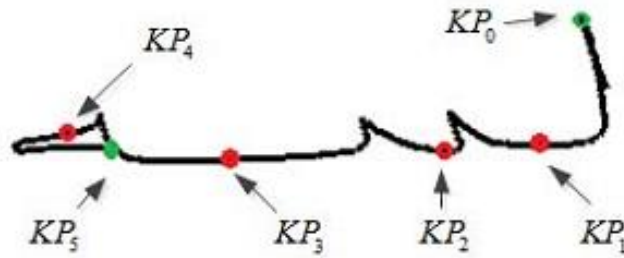


Figure 2-12 The Key Points of word لبيه [33]

Different segmentation algorithms for online handwritten Arabic text were discussed by Abuzaraida et al. in 2010 [34].

2.6 Feature extraction techniques

Extracting good and representative features is very important in any recognition system as it contributes to the recognition performance. Moreover, selecting a proper feature is considered as the most important factor influencing the recognition accuracy [35]. Selecting the appropriate features is based on many factors such as the text nature, the type of system (i.e. offline or online) and the type of the script (i.e. printed or handwritten). In the literature, a number of approaches have been used in modeling Arabic online handwritten text which may be word, character or digit. These approaches aim to extract the most pertinent features of the input that achieve better classification. In general two main approaches are available: structural and statistical approaches. In the following paragraphs we will review the research work in these approaches.

2.6.1 Structural Features

Structural features are a set of perceptual entities that describe the geometrical characteristic of text such as loops, dots and intersection points [36]. The challenge with structural feature is that it requires a robust algorithm to extract the primitives, which is not always assured. [37].

Arabic text has different structural features that can be extracted from the structure of the text's shape like loops, end points, straight lines etc. Figure 2-13 shows some structural features in an online text.

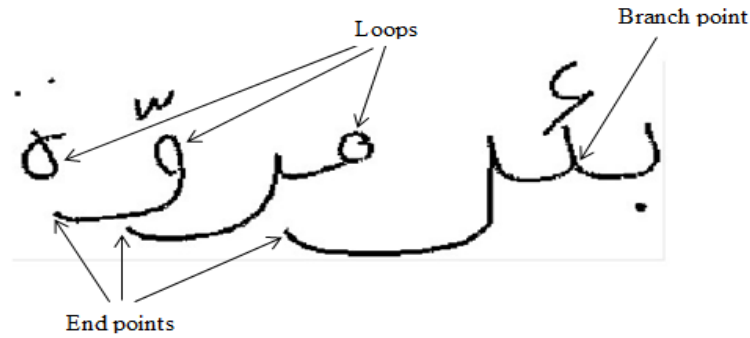


Figure 2-13 Illustrations of some structural features.

These structural features can be used to differentiate between the different characters. Structural features are considered to be more common for the recognition of Arabic script compared to Latin script [38].

An early structural approach for Arabic offline cursive handwritten words was proposed by Almuallim and Yamaguchi in 1987 [39]. In their approach, words are segmented first into a set of strokes which are defined as a continuous curve represented by a string of (X_i, Y_i) . For each stroke, the following information is extracted: The start and end points, the stroke length, the frame of the stroke (i. e. X_{min} , X_{max} , Y_{min} , Y_{max}) and the connection point with the previous stroke. Based on this information, the stroke is classified into one of five groups which are: dots, Hamza, strokes that contain loops, strokes without loops and connected with the next stroke and strokes without loops and end with a line end. A recognition rate of 81.25% was reported on a database of 400 words. However, these words are collected from two writers only and written in a formal writing style, which is not the case in practical applications.

In 1990, Al-Emami and Usher proposed a structural approach for Arabic online words based on decision tree techniques [40]. After finding the segments of the word, each

segment is categorized into one of four directions based on its direction. The following information is also extracted for each segment: the segment's length, flags referring to dots and a tangent value representing the slope. A decision tree then is used to store the information on the segments in the learning process, and for the search to find a segment in the recognition process. The reported recognition rate was 100% for writer dependent. However, when the tested words were written by one writer who did not provide data in the training phase the achieved recognition rate drops to 86% (43 words out of 50 word were correctly recognized). The samples used in this work are limited to the letters of four words of a postcode. These words comprise only ten different letters which may explain the high recognition rate.

An early structural approach for recognition online Arabic handwritten digits was proposed by Beigi et al. in 1994 [41]. In their approach, a five dimensional feature vector is used (the difference between neighboring coordinate, the sine and cosine at each point and the absolute y-coordinate of each point shifted by the computed baseline value). Hidden Markov models (HMMs) was used in the classification phase. For training, 6000 samples of Arabic digits written by 20 different writers were used whereas 700 samples written by 14 writers were used for testing. The reported recognition rate was 93.14%. The proposed system used single state HMM and the reported accuracy may be increased by using multiple-state HMM which would model the duration of each part of the digit much better.

Trajectory/velocity modeling approaches were applied in several works. Kherallah et al. [42] proposed a neural network system to recognize online Arabic digits based on Beta-circular approach. Their approach represents the first work that combine the geometry and

the kinematics in the trajectory modelling. Each stroke is represented by dynamic Beta and static circular parameters. These parameters are extracted from the curvilinear velocity of the points. In their later work [43] they improved the modeling quality of handwriting by introducing elliptical parameters and using velocity signal extremum to estimate the number of strokes. They did not use the overlap of the Beta signal which presents an important factor to determine the number of strokes. According to Beta–elliptic representation used in this system, the number of features for each character may reach 63. To reduce the problem of dimensionality, the authors proposed a new approach for modeling the handwritten trajectory. Their approach is based on inflection point detection, the overlapped form of beta signals, and the elliptic arcs [24]. The trajectory is modeled using velocity-based (Beta function parameters) and trajectory-based (elliptic parameters) features where the stroke is represented by seven values. The first four values (Beta parameters) describe the global timing features whereas the last three values (elliptic parameters) reflect the global geometric features.

Al-Taani et al. proposed a feature extraction approach for recognizing of handwritten Arabic digits based on changing signs of the slope values [44]. In his algorithm, the slope is estimated and normalized for adjacent nodes. Each digit is represented by a set of primitives which are identified and extracted based on the changes of the signs of the slope values. These primitives are represented by a string which is a production of a grammar. In order to identify the digit, a special grammar is used to match the string of primitives to the corresponding digit. A dataset collected from 100 writers is used to evaluate the proposed algorithm and the reported average recognition rate was about 95%.

In 2012, Abdul Azeem et al. proposed an approach that uses both structural and statistical features to recognize Arabic online digits [14]. The statistical features were extracted by converting the user's trajectory into a bitmap image. The image's size then normalized to 30X30 pixels. A database containing 30,000 digits were collected from 300 users to test the proposed approach. The overall reported recognition rate was 98.73%.

2.6.2 Statistical Features

Statistical features are used in many recognition systems due to their reliable results; however statistical approaches require a large amount of data for training [19]. On the other hand, statistical approaches offer high speed with large memory compared to structural approaches [45]. Normally, statistical features are based on numerical measures that are computed over images or regions of images. There are many machine learning paradigms, such as Hidden Markov Models (HMM) and Support Vector Machine (SVM), available for classification based on statistical features.

In 2002, an online system for the recognition of handwriting Arabic characters was proposed by Mezghani et al. [46]. Fourier descriptors were used in this study as a feature vector to represent the online Arabic characters. In their later work, the authors improve their work by combining two Kohonen maps using two different character representations [47]. These character representation techniques are Fourier descriptors and tangents extracted along the characters online signals. Then, they use majority voting decision rule to combine the two Kohonen maps results and favors the most reliable one. In another recent work Mezghani et al. [15] represent the characters by empirical distribution (histograms) of tangent differences at regularly sampled points on the characters signal

[15]. This representation is used to investigate Bayes classification for online Arabic characters using Gibbs modeling of the class-conditional density functions. They estimate the parameters of Gibbs density functions according to constrained maximum entropy formalism proposed by Zhu et al. which was originally used for image and shape synthesis [48].

In 2008, Izadi and Suen defined a new statistical feature called Relative Context (RC) which is extracted from the relative pairwise distances and the angles on the trajectory of a digital character [45]. All pairs of points on the trajectory are used in order to preserve local descriptors in the context of the global character shape. Their approach was tested with a database of Arabic isolated characters containing 528 samples. The reported recognition rate was 97.8%. The authors claim that this performance outperforms the best accuracy reported in literature on the same database.

The Hu's moments were adopted by Daifallah et al. for extracting the features of online Arabic hand-written words [22]. Hu's moments, which are a set of seven compound moments, are considered as an offline features extraction approach. In this approach, each character is featured by Hu's seven moments, $h1$ to $h7$, which are invariant in the continuous image domain to translation, rotation, and scale change. The reported recognition rate was about 92% for words and 97% for letter recognition. These recognition rates were obtained with the same training user. When the system was tested with new users, the recognition rates dropped to 71% and 79% for words and letters, respectively. This work was evaluated using 150 words without dots or marks (above or below the words).

Boubaker et al. proposed an online Arabic handwriting modeling system based on fuzzy graphemes segmentation [32]. To overcome the problem of crisp segmentation points, a fuzzy segmentation is adopted by associating to each candidate point a fuzzy degree of confidence as follows: after detecting the baseline, fuzzy confidence degree is estimated and associated with each vertical extremum trajectory point to be a particular point of segmentation. Therefore, the fuzzy confidence degree is associated with two types of candidate segmentation points: angular points and bottom of valleys. The segmented fuzzy graphemes are described using Fourier descriptors parameters. The proposed model is evaluated on the ADAB database of 937 online Tunisian city names which consists of 15158 samples using a Hidden Markov Models classifier. The reported results show that the recognition rate is improved using the fuzzy segmentation approach compared to the crisp one. Table 2-4 shows a summary of some techniques used for feature extraction for Arabic online text recognition.

Table 2-4 Summary of some techniques used for feature extraction for Arabic online text recognition

| Author(s) | Features | Features' Type | Scope |
|------------------------------|---|----------------------------|--------------|
| Beigi et al. 1994 [41] | Geometric features | Structural | Digits |
| Kherallah et al. 2002 [42] | Circular and Beta features | Structural | Digits |
| Kherallah et al. 2004 [43] | Elliptical and Beta features | Structural | Digits |
| Kherallah et al. 2008 [24] | Elliptic parameters and Beta function parameters | Structural | Digits |
| Al-taani 2008 [49] | The changes in the slope's values + The primitives' string | Structural | Digits |
| Abdul Azeem et al. 2012 [14] | Temporal (online) and spatial (offline) features | Structural /Statistical | Digits |
| Mezghani et al. 2002 [46] | Elliptic Fourier descriptors | Statistical | Characters |
| Mezghani et al. 2003 [47] | Fourier descriptors and tangents | Statistical | Characters |
| Izadi and Suen 2008 [45] | Relative Context (RC) feature | Statistical | Characters |
| Mezghani et al. 2008 [15] | Histograms of tangent differences | Statistical | Characters |
| Omer and Ma 2010 [50] | Freeman code | Structural | Characters |

| | | | |
|-------------------------------|--|-------------------------|--------------------|
| Al-taani and Al-Haj 2010 [21] | Structural features that include: number of segments, loop, sharp edges, secondary segments, similarity of secondary segment, horizontal–vertical Orientation | Structural | Characters |
| Khodadad et al. 2011 [51] | Spatial features such as loops and stroke directions and temporal features based on Discrete Cosine Transform of the trajectory coordinates | Structural /Statistical | Characters |
| Ismail and Abdullah 2012 [27] | Two type of features are used: Edge Direction Matrixes (such as homogeneity and edges regularity features) and geometrical features (like the width of the dot, number of occurrence in horizontal and vertical projection). | Structural /Statistical | Characters |
| Kour and Saabne 2014 [52] | Shape context feature and the Multi Angular Descriptor (MAD) | Structural /Statistical | Characters |
| Daifallah et al. 2009 [22] | Hu’s seven moments | Statistical | Characters / Words |
| Sternby et al. 2009 [31] | Structural features such as angles, length ratio, arc type, and the relative positions | Structural | Characters / Words |
| Al-Emami and Usher 1990 [40] | Segments’ directions | Structural | Words |
| Kherallah et al. 2009 [53] | combine visual encoding, beta-elliptical model | Structural | Words |
| Boubaker et al. 2010 [32] | Fourier descriptors, Fuzzy segments | Statistical | Words |

2.7 Classification approaches

Several classification approaches have been discussed in connection with the feature extraction techniques described in the previous section. In addition to these approaches, many other classification approaches have been used by researchers for the recognition of Arabic Online digits, characters, and text. These classification approaches include Hidden Markov Models [54] [55] [56] [57], Artificial Neural Networks (ANN) [51] [42] [58], Support Vector Machines (SVM) [18] [14], k-Nearest Neighbors (k-NN) [59], dynamic programming [60] [61], decision trees [21] [50], template matching [31] [62] and combination of different approaches [24] [27].

Statistical classifiers, such as Hidden Markov Models and Artificial Neural Networks, are used in many recognition systems due to their reliability. Artificial Neural Networks with three layers feed forward is used by Khodadat et al. in 2011 for Arabic/Persian character recognition system [51]. They represent the characters using Discrete Cosine Transform (DCT) of their x and y coordinates. More than 3000 characters are used for training and testing the system and the reported recognition rate is 95.69%.

An HMM/NN hybrid system was proposed by Tagougui et al. in 2013 where the discriminative powers of the neural network and Markovian sequence modeling are utilized [19]. The input trajectory is first segmented into small and continuous parts called segments then the features of these segments are extracted based on the enhanced Beta-Elliptical strategy [63]. These segments are used for training the neural network and the output of this neural network is decoded by Hidden Markov Models to generate character level

recognition. For the training phase, segmented characters are used by segmenting 6000 words chosen randomly from sets 1, 2 and 3 of ADAB database. The system was tested on sets 4, 5 and 6 of ADAB and the reported accuracy was 96.4%.

Kherallah et al. proposed an approach that combine visual encoding, beta-elliptical model and genetic algorithm for recognition of Arabic online words [53]. The proposed approach is evaluated using a dataset of words selected from ‘‘LMCA’’ database. This dataset includes 500 words written by 24 writers. The average of obtained recognition rate is 97%.

The application of a template matching scheme is explored in the recognition of Arabic online text by Sternby et al. [31]. The proposed approach is based on an additive single character recognition method [64]. Their approach starts by segmenting the text into a set of segments representing at most the shape of one individual character. This segmentation is based on the vertical extreme points with respect to the writing direction and a set of heuristic rules. Each segment is then represented by the features angle (ϕ), connection angle (θ), arc type (T), length ratio (λ), and the relative positions R_x , R_y . The graph strategies proposed for connected character recognition are used for matching segments from the template database to the segments of the input text [65]. The proposed approach is tested on a set of 1,578 words collected from 40 writers. The reported recognition rate is about 91% for word recognition.

In order to reduce the search space, a hierarchical Arabic online recognition system was proposed by Saabni and El-Sana in 2009 [62]. In their work, they avoid segmenting the words into individual characters by adopting a holistic approach. The final recognition decision is taken based on several filters in a hierarchical manner. The number of candidate

words are reduced in the first filter using global features and delayed strokes. In the second filter, a modified dynamic time warping (DTW) classifier, which is considered as a template matching classifier, is used with local features to measure the similarity between the input word and all candidate words. In the last filter, shape context features are used to compare the top k ranked word-part against the written word-part. The reported recognition rates reaches 90% after using shape-context based classifier. However, the quadratic time and space complexity represents the main drawback of DTW classifier [66].

Dynamic Time Warping (DTW) was used also by Kour and Saabne in 2014 to re-score the candidate characters returned from k-NN classifier [52]. Preprocessing steps followed by extracting shape context features [67] and multi angular descriptors [68] are used to generate the characters' feature vectors. Then an embedding of the feature vectors into a normed wavelet coefficients domain is employed in order to allow fast classification. Consequently, Earth Movers Distance metric is approximated using the Manhattan distance. Fast retrieval of the k nearest neighbors, potential letter candidates, for a testing character was then possible using k-NN. DTW was used to refine the similarity scoring of the candidates. In their work, they manually segmented some words from ADAB database to extract a set of characters for both training and testing. The extracted set comprises 5602 characters and covers the different forms of the characters. The reported accuracy was 91%.

Elanwar et al. adopted dynamic programming technique for developing an Arabic online character recognition system [60]. They used dynamic programming for computing the Minimum Edit distance between direction features of the testing data with the skeleton patterns built during the training phase. Their system is trained using 1814 characters

collected from four writers. The testing data includes 435 characters collected from other four writers. The reported recognition rate is 95%.

Dynamic programming is also used by Abuzaraida et al. to recognize Arabic online digits [61]. In their system Global Alignment Algorithm (GAA) is used as recognition engine to recognize the Arabic digits. The digit trajectory directions are taken as the main features for their system by using freeman chain code to find the direction matrix for each digit. The system is tested using a dataset collected from 50 writers where 150 samples were collected for each digit. An average of 98% accuracy was archived using 80% of the collected digits for training and 20% for testing.

Decision tree is used with the structural features to recognize the Arabic online characters by Al-Taani and Al-Haj in 2010 [21]. The proposed decision tree is split into four subtrees due to the different features attached to the different classes of characters. For the input character, the values of the features determine the branch that should be selected. The system fails to recognize the character if its feature values are not labeled on any branch of the tested features. The proposed system is evaluated on a set of 1,400 isolated characters written by ten users where each user wrote the 28 Arabic characters five times. The reported recognition rate was 75.3% for all characters. Their experimental results show that the proposed approach did not perform well on the characters that contain sharp edges.

A recognition system for Arabic online characters based on a combination of matching algorithm and decision tree was proposed by Omar and Ma in 2010 [50]. In their system, the characters are divided into four sets based on the number of dots. Freeman code is used to represent the characters where every character is represented by a string of directions.

A decision tree is used to classify an unknown character into one out of four classes according to number of dots. Then matching algorithm is used to find the similarity between the directional stroke string of the unknown character and the strings' database. Self-collected data was used in the training phase where four writers are invited to write 336 samples of isolated Arabic character set. Other dataset comprising 140 characters was collected from five writers to test the system. The average reported recognition rate was 97.6%. The approach is tested with the isolated forms of Arabic characters without considering the beginning, middle or end forms.

A rule based approach for Arabic online character recognition was proposed by Ismail and Abdullah in 2012 [27]. Twenty eight production rules (i.e. IF-THEN rules) are built for the twenty eight Arabic characters. These rules are built based on the hybrid Edge Direction Matrices (EDMS) [69] and geometrical features extracted from the characters. In addition to the production rule classifier, artificial neural network and decision trees are also used in order to achieve reliable results. They used 504 characters in the training phase and 336 characters in the testing phase. These characters are collected from different users. The achieved accuracy was 97%. The proposed approach works only on isolated Arabic characters.

A sequential version of multiple classifiers is used by Kherallah et al. for recognition of Arabic online digits [24]. In the learning phase, an association of the Self-Organization Maps (SOM) with Fuzzy K-Nearest Neighbor Algorithms (FKNNA) are used with Beta signals and Elliptic arcs features. The result of the learning phase is used in the training process of Multi-Layers Perception Neural Networks (MLPNN) classifier which is used in

the classification process. The MLPNN classifier output represents the membership degree of the testing digit to the digits' classes. To test the performance of the proposed system, a dataset comprising 30,000 digits is collected from twenty four writers where 20,000 was used in the learning process and the rest was used for testing the approach. The reported recognition rate was 95.08%. Table 2-5 summaries some classification techniques for Arabic online text recognition.

Table 2-5 Summary of some classification techniques for Arabic online text recognition

| Author(s) | Classifiers | Data | | Accuracy | # Writers |
|----------------------------------|---|--------------------|-------------------|----------|------------|
| | | Training | Testing | | |
| Mustafa et al. in 2015 [61] | Global Alignment Algorithm (dynamic programming) | 1200 digits | 300 digits | 98% | 50 writers |
| Kherallah et al. 2008 [24] | MLPNN developed in a fuzzy concept | 20000 digits | 10000 digits | 95.08% | 24 writers |
| Elanwar et al. [60] | Dynamic programming technique | 1814 characters | 435 characters | 95% | 8 writers |
| Khodadad et al. in 2011 [51] | Artificial Neural Network | 2800 characters | 560 characters | 95.69%. | -- |
| Al-Taani and Al-Haj 2010 [21] | Decision tree | 1400 characters | | 75.3% | 10 writers |
| Omar and Ma in 2010 [50] | Decision tree and matching algorithm | 366 characters | 140 characters | 97.6% | 9 writers |

| | | | | | |
|---------------------------------------|--|--|---------------------|-------------|-------------|
| Ismail and siti Abdullah in 2012 [27] | Production rules/ Decision Tree/ Neural Network | 504 characters | 336 characters | 97% | -- |
| Kour, G. and Saabne, R. in 2014 [52]. | Earth Mover's Distance (EMD) and Dynamic Time Warping (DTW) | 5602 segmented characters from ADAB database | | 91%. | --- |
| Tagougui et al. in 2013 [19] | Hybrid HMM/NN | ADAB Sets 1,2 and 3 | ADAB Sets 4,5 and 6 | 96.4% | 130 writers |
| Saabni and El-Sana 2009 [62] | Modified Dynamic Time Warping (DTW) and shape-context based classifier | -- | 1000 PAWs | 86% -90% | 10 writers |
| Kherallah et al. 2009 [53] | Genetic algorithm | 500 words from "LMCA" database. | | 97%. | 24 writers |
| Sternby et al. 2009 [31] | Template matching | 1,578 words | | 91% | 40 writers |

CHAPTER 3

FUZZY MODELING FOR ARABIC ONLINE DIGITS

RECOGNITION *

In this chapter we present a novel fuzzy technique for Arabic (Indian) online digits recognition. We use directional features to automatically build generic fuzzy models for Arabic online digits using the training data. The fuzzy models include the samples' trend lines, the upper and lower envelopes of the samples of each digit and the automatically generated weights for the different segments of the digit models are also used. In addition, the fuzzy intervals are automatically estimated using the training data. These fuzzy models proved to be able to handle the variability of the handwriting styles. The classification phase consists of two cascaded stages, in the first stage the system classifies digits into zero/nonzero classes using five features (viz. length, width, height, height's variance and aspect ratio) and the second stage classifies digits 1 to 9 using fuzzy classification based on directional and segment histogram features. SVM is used in the first stage and syntactic fuzzy classifier in the second stage. A database containing 32695 Arabic online digits is used in the experimentation.

* A paper is accepted based on this chapter

3.1 Introduction

In recent years much research has been conducted on the recognition of offline [70] [71] [72] and online [73] [26] Latin digits. Most of the research on Arabic digits addressed offline Arabic handwritten digits [8] [74] [75] [76] [77] [78][79] [80] while few addressed online Arabic digits [14] [24] [41] [42] [43] [49].

Automatic recognition of handwritten Arabic (Indian) digits has a variety of applications including banking systems and forms filling. Although printed/handwritten Arabic text is cursive, Arabic numerals are not cursive and hence segmentation of individual digits may not be needed unless the digits are touching. Figure 3-1 shows samples of handwritten Arabic digits from 0 to 9 along with their printed versions.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ٩ | ٨ | ٧ | ٦ | ٥ | ٤ | ٣ | ٢ | ١ | . |
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | . |

Figure 3-1 Arabic (Indian) digits 0 to 9

In this chapter we present a novel technique for the automatic recognition of Arabic online digits. In this technique, automatic fuzzy modeling of Arabic online digits is implemented. We automatically generate fuzzy models of the different digits using the segments' directions of the Arabic online digits using the training data. The models include the samples trend line of each digit and the digits' model envelopes. While the skeleton (generated using offline Arabic character clustering) is used to define the character models in [81] and models were not used (only fuzzy similarity measure is used) in [82]. In this work, we automatically generate the fuzzy intervals based on the analysis of the training samples and they are not set manually as in [81] and [82]. In addition, our proposed

technique automatically generates weights for the different segments using the training samples. These weights are integrated in the fuzzy models while weights were not used in [82] and the segments are assumed of equal weight in [83]. These weights improve the recognition rates because they are assigned based on the importance of the different segments of the digit.

For the classification, we use a two stage approach where SVM is used in the first stage to classify digits into zero and nonzero (using statistical features) and fuzzy-based approach is used in the second stage to classify nonzero digits (digits 1 to 9) using the automatically generated models in the second stage. The second stage has an integrated feedback verification step which verifies the recognized test sample label. If the first label does not pass validation (using other features) then the next label in the list is selected in the feedback loop and so on. Figure 3-2 shows the overall block diagram of the implemented system.

The contribution of our approach automatically generates robust fuzzy models for Arabic online digit recognition using the training data. The models include the trend lines, the upper and lower envelopes of the samples of each digit. The fuzzy durations are automatically generated and are set at the digit segment level. In addition, automatic generation of weights at the segment level, which indicates the importance of the different segments of the digit, was integrated into the model. This is an improvement over previous work [81], [82] and [83] in several aspects (viz. automatic model generation over both [81] and [82] ; weights of the different segments were not used in [82] and [83] assuming equal weights; only similarity estimate is used in [82] and no fuzzy modeling; Parvez et al.

applied it to the skeleton of the Arabic offline characters [82] and Halawani applied it to the polygonal approximation of the Arabic online characters [83] and here we apply it to Arabic online digit recognition). A fuzzy syntactic classifier is implemented with integrated feedback stage that verifies the selected classes of the test samples using segment histogram features. This stage improved the accuracy of the syntactic fuzzy classification.

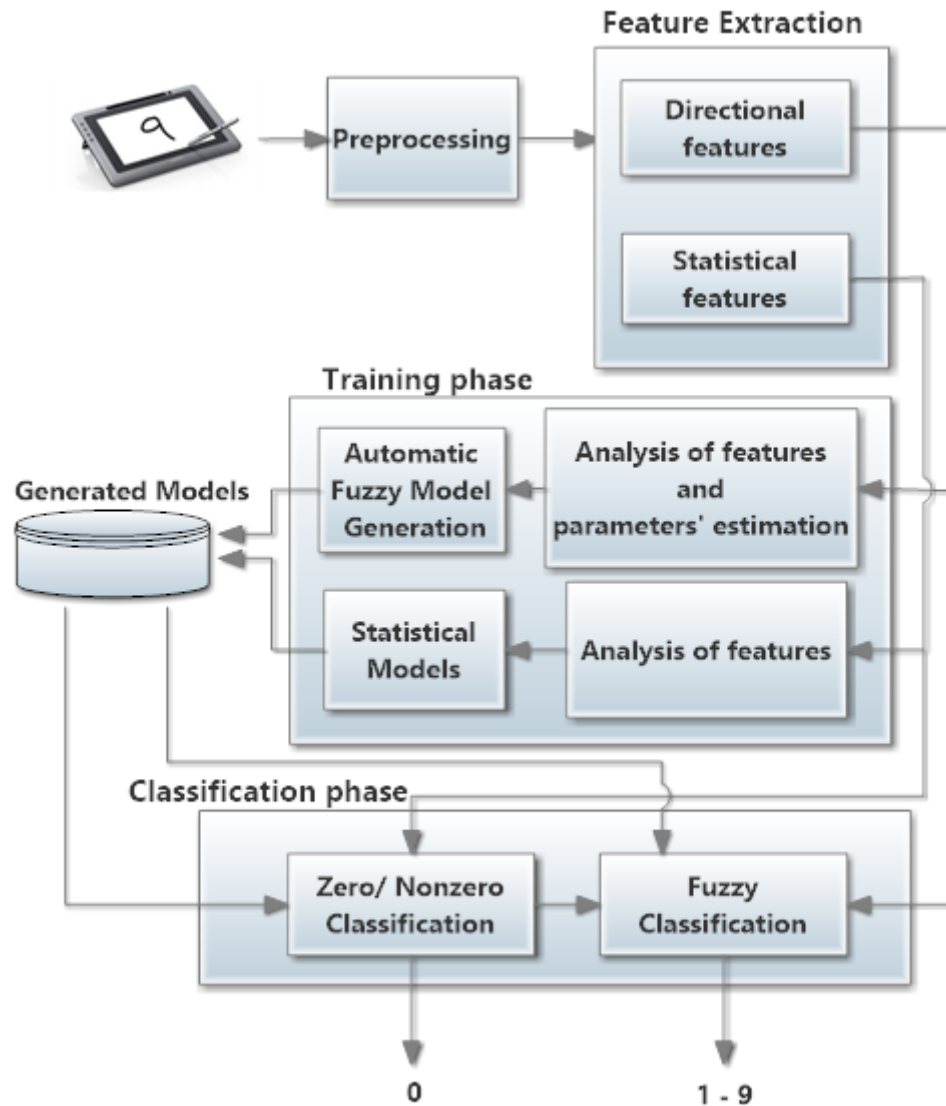


Figure 3-2 Overall block diagram of our approach for Arabic online digit recognition

The rest of the chapter is organized as follows. Section 3.2 presents the preprocessing techniques applied to Arabic online digits. Features used in this work are detailed in section 3.3; section 3.4 describes the proposed analysis and automatic fuzzy model generation approach; the classification phase is discussed in section 3.5; the experimental results are reported in section 3.6; and finally the conclusions are presented in section 3.7.

3.2 Preprocessing

Writing using keyboard-less devices is less controlled than writing using a pen on paper. Therefore, data collected by using these devices is affected by hardware imperfections and the trembles in writing. Preprocessing is crucial to achieve better recognition rate [84]. In this work, simplification, smoothing and normalization are applied to the data before feature extraction.

Douglas-Peucker algorithm [85] is used to simplify the curves in the digits by reducing the number of points representing the curves. The algorithm removes unimportant points from the curve to simplify it as shown in Figure 3-3.

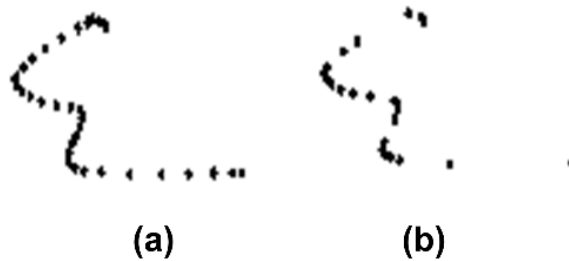


Figure 3-3 Digit four (a) before simplification (b) after simplification

An algorithm to remove the small curve variations and noise is applied. Any small segment at the beginning or at the end is considered as noise. In addition, if there are two consecutive

segments that are very close to each other but having different directions, the first one is considered as a noise. The following figure shows digit 6 before and after preprocessing.



Figure 3-4 Digit six (a) before preprocessing and (b) after Preprocessing

In this work we also normalize the data by adjusting all the samples' lengths to 25 points. This length is derived based on the common lengths of samples. We experimentally found that this length is enough to model the samples.

3.3 Feature Extraction

Extracting good and representative features is very important in any recognition system as it contributes to the recognition performance [86]. Three sets of features are extracted in this work. Shape features are extracted first to classify digits into zero or nonzero. Then, directional and histogram-based features are used to recognize nonzero digits (digits from one to nine).

3.3.1 Shape Features

Unlike other digits, Arabic digit zero is just like a dot. The way it is written when magnified have different shapes as shown in figure 3-5. These shapes are sometimes confused with other digits like 1, 5, 8, etc. A human seeing the samples in figure 3-5 will not be able to identify them as zeroes.



Figure 3-5 Different samples of Arabic digit zero ‘0’

Due to these difficulties, we address the ‘zero’ digit separately based on its length, width, height, heights’ variance and aspect ratio (height-to-width ratio).

3.3.2 Directional Features

Each digit is represented by taking the directional information of all points representing that digit. Let $P = (x_i, y_i)$, where $i=1,2,\dots,n$ be the sequence of points that represent digit D . The directional features are $\Theta_D = [d_1, d_2, \dots, d_{n-1}]$, where d_i is the angle between the points (x_i, y_i) and (x_{i+1}, y_{i+1}) .

3.3.3 Histogram-based Features

Basically, each digit consists of a set of segments and each segment has its own orientation. In our histogram-based feature, segments’ orientation takes values from a finite set of orientations or directions called standard writing directions. Standard directions can be regarded as quantization of stroke directions. Figure 3-6 (a) shows the standard writing directions with 22.5° gap (i.e. number of directions is sixteen). The proposed histogram-based features are based on the histogram of digit segment orientation. Let $\text{Seg}_i = 1, 2, \dots, n$, be the segments of a digit D . The histogram features vector $S_D = [P_1, P_2, \dots, P_{16}]$, where 16 represents the number of directions used in our work, and P_j is the percentage of the segments of digit D in direction j such that:

$$P_j = \frac{\text{Number of segments of direction } j}{\text{Number of all segments}}$$

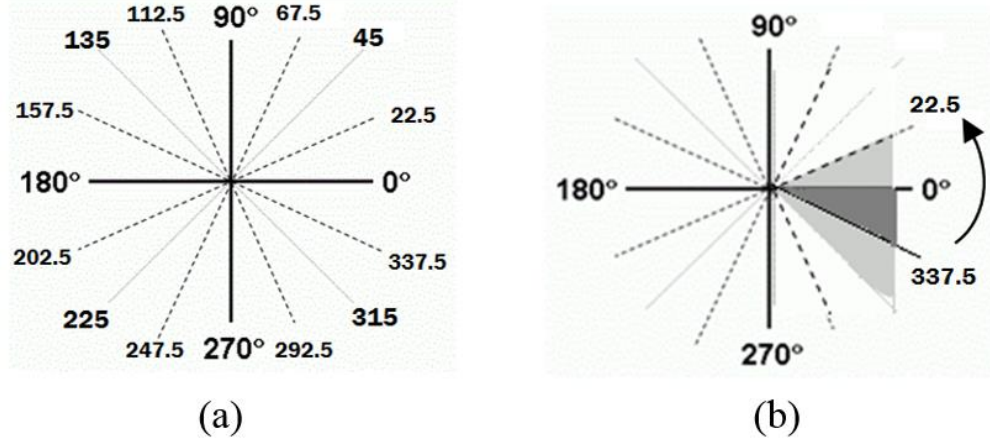


Figure 3-6 (a) Standard writing direction with 22.5° gap (b) The overlap between direction 337.5° and direction 0°

Due to the variations in writing style, the computation of the segments' directions should tolerate some level of variations in writing directions. To alleviate this problem, we represent each direction i by $i \pm 22.5^\circ$ (i.e. direction i represents all segments having directions from $i - 22.5^\circ$ up to $i + 22.5^\circ$). For example, direction 0° includes all segments having directions from -22.5° (337.5°) up to $+22.5^\circ$. Accordingly, there will be an overlap of 22.5° between consecutive directions' segments. The overlap between direction 337.5° and direction 0° is shown in figure 3-6 (b). Figure 3-7 (a) shows a sample of digit 'six' with the directions of its segments; the histogram vector values for this sample are $[0.02, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.37, 0.39, 0.02, 0, 0.20]$ and its histogram is shown in figure 3-7 (b).

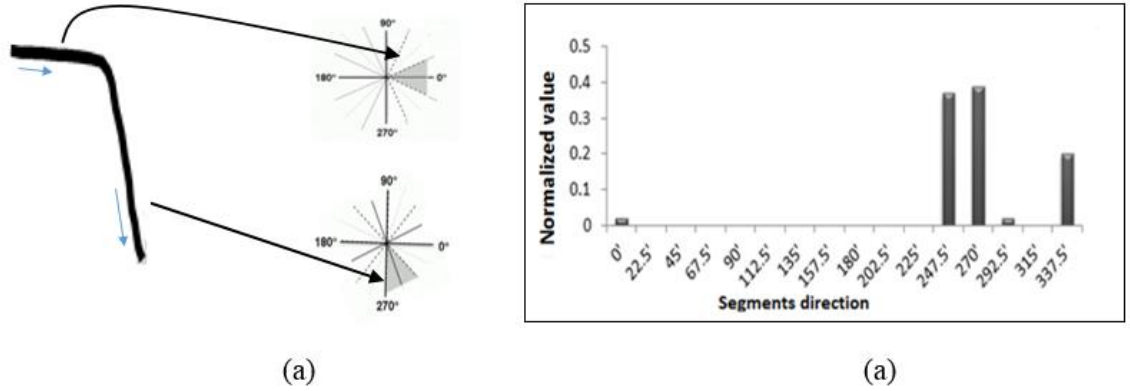


Figure 3-7 (a) Sample of digit six '6' with the directions of its segments (b) Segments histogram

3.4 Automatic Fuzzy Models' Generation

In this section we present the generation of fuzzy models to address the problem of variability in writing.

3.4.1 Automatic Fuzzy Modeling

The directional feature representation of digits reflects their shapes. This representation is used to build generic models that represent the digits (i.e. fuzzy models). In order to generate the fuzzy models of digits, the lengths of all digit samples are normalized to a unique length. In this work, we normalized all the samples' lengths to 25. This length is derived based on the common lengths of the samples. After normalizing the samples, the directional features of the training samples are extracted. The length of the directional features' vector is 24 since it represents the sequence of angles between each two consecutive points. Figure 3-8 shows the directional representation of several samples of digit 'three' written by different writers. We implemented two approaches; Model Trend Line (MTL) and Model Envelope (ME) [83]. MTL is a single trajectory that represents the mean of all samples. ME uses the top and bottom envelopes of the aggregated samples. Top

and bottom envelopes of the samples form a good model to describe the shape of the digits' models.

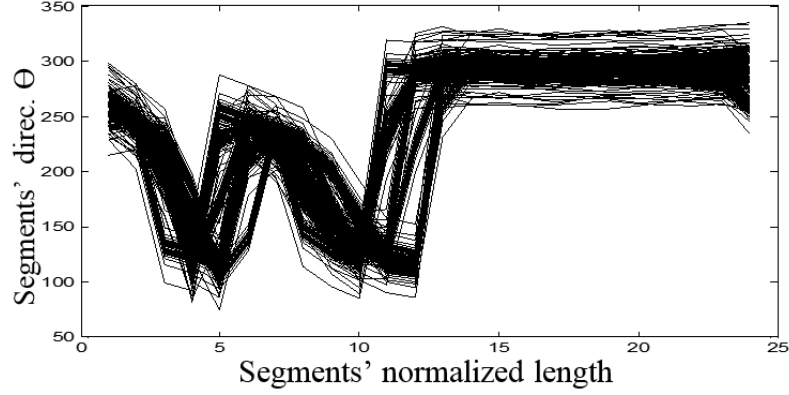


Figure 3-8 Directional representations of several samples of digit three

3.4.2 Model Trend Line (MTL)

The Trend Line (TL) is a trajectory that represents the center of the aggregated samples and hence the general shape of the samples of the classes. The aggregation is done by estimating the mean at each line segment (sampling point):

$$TL_m(s) = \frac{1}{n_s} \sum_{i=1}^{n_s} y_i$$

Where s is the line segment index, n_s is the number of points in segment s , y_i is the segments' direction at point i of the segment. The standard deviation is given by the following equation:

$$TL_s = \sqrt{\frac{1}{n_s} \sum_{i=1}^{n_s} (Y_i - TL_m(s))^2}$$

This representation shows the general shape of the digit. Figure 3-9 shows an example of the MTL for digit three.

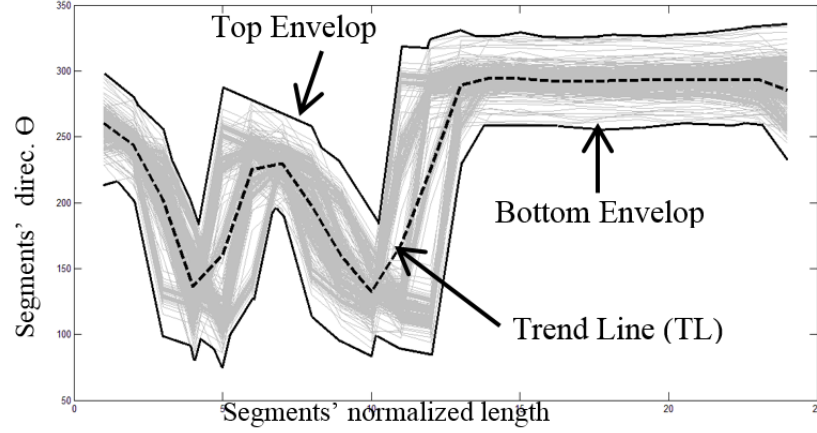


Figure 3-9 MTL and Model Envelop for digit three

3.4.3 Model Envelop (ME)

ME is represented by two curves that surround the main body of the samples aggregation. The top and bottom envelopes describe the range and shape of each digit main body. Every sample is represented as a function $y = C(x)$. Given a set $C = \{C_1, C_2, \dots, C_n\}$ of x -curves that represent n samples. The bottom envelope is defined as the point-wise minimum of all samples. The bottom envelope for the set C at position i can be defined as follows:

$$B_c[i] = \min C_i(x),$$

where $x=1, 2, \dots, n_s$, and $i=1, 2, \dots, N$, is the number of points in each sample. Similarly, the top envelope of C is the point-wise maximum of the samples curves in the set:

$$T_c[i] = \max C_i(x),$$

where $x=1, 2, \dots, n_s$, and $i=1,2,\dots,N$. Figure 3-9 demonstrates the aggregated samples that form digit ‘three’. The top envelop, shown in bold, represents the upper limit of the model and the bottom envelop, shown in bold, represents the lower limit.

In our experiments the fuzzy model generated for each training sample is a series of points that represent the directional features of that sample with the standard deviation (σ) of y-values of all the training samples from the same class at each point. The standard deviation for each class (digit) is calculated from the training samples of that class. Figure 3-10 shows a fuzzy model for digit ‘three’ which is generated from the directional representations of the training samples of this digit.

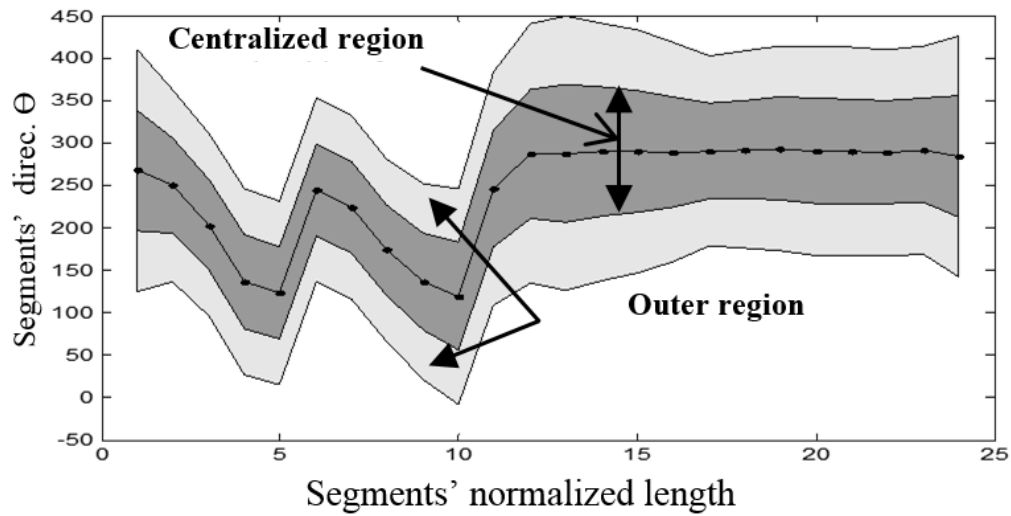


Figure 3-10 Generated Fuzzy models for digit ‘Three’

The shaded area in Figure 3-10 represents the fuzzy tolerance of the model. The width of this area on both sides is related to the standard deviation which is estimated in the training phase. The width of the fuzzy tolerance varies from one point to another according to the model as estimated in the training phase. Figure 3-11(a) shows the directional

representation of several samples of digit ‘Six’ written by different writers. Figure 3-11(b) shows a sample of digit ‘Six’ mapped to the fuzzy model generated for these samples of digit ‘Six’.

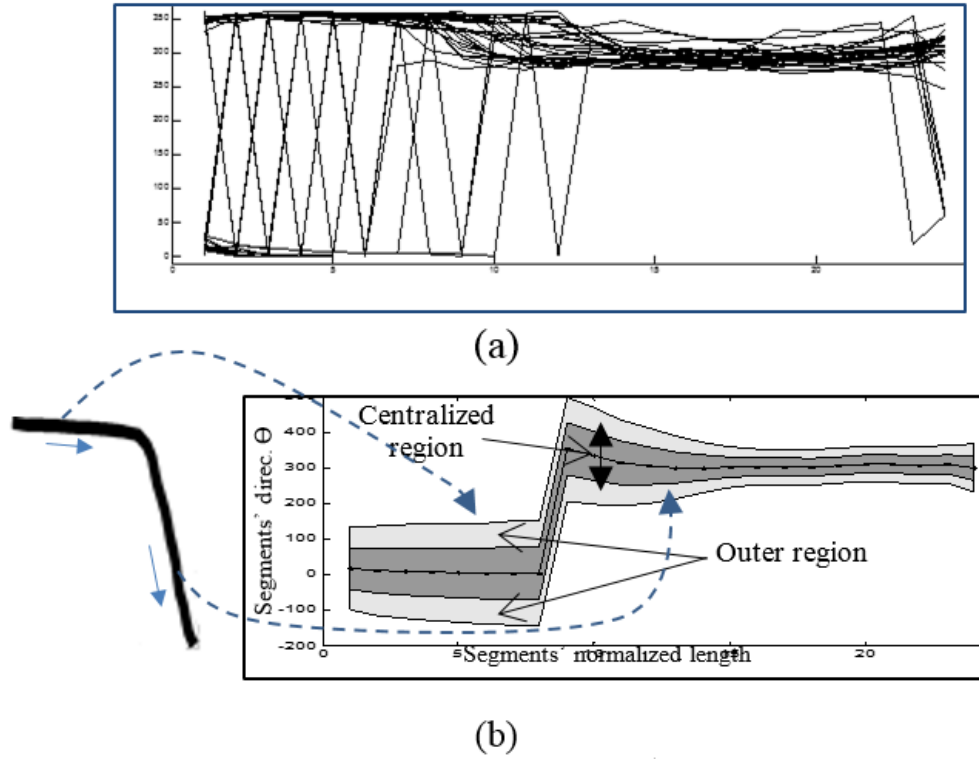


Figure 3-11 Generated Fuzzy models for digit ‘Six’

3.5 Classification

The proposed classification technique has two stages, namely Zero/Nonzero and Fuzzy classification. The first stage recognizes digit zero ‘0’ and the second stage recognizes digits one ‘1’ to nine ‘9’.

3.5.1 Zero/Nonzero Classification

In this stage, the shape features are used to classify digits into zero and nonzero. Support Vector Machine (SVM) classifier with Radial Basis Function (RBF) as the kernel was used. In this classification, we use the length, width, height, height variance and aspect ratio of the digit as features.

3.5.2 Fuzzy Classification

This stage is used to classify nonzero digits (digits 1 to 9). Directional and histogram-based features are used in this stage. Directional features' vector of each test sample is compared with all fuzzy models to find the most similar one.

As a result of the normalization process, the directional features of each digit are represented by 24 features (angles). At each point, fuzzy comparison between the sample and the models is done to estimate their similarity. Figure 3-12 illustrates the fuzzy comparison between a sample and a model. All the sample and model points are aligned where the testing sample curve is marked by plus points and circular points for the model line. The shaded area in Figure 3-12 represents the fuzzy tolerance of the model. The width of this area is based on the standard deviation estimated in the training phase, and it is different from one point to another.

The membership of any point outside the tolerance region (shaded area) is considered as zero. For the shaded area, the membership of any point in the middle area (dark area) is taken as 1 which represents the peak of a fuzzy set. The membership outside this area (light area), M_v , is calculated based on Equation (1) and trapezoidal membership function as

shown in Figure 3-12. Note that we are using membership functions based on automatically generated duration and not fixed as in [82]. When computing the membership value at a certain sampling point, the value of the membership is determined based on one of four cases as follows; the membership value is denoted by M_v .

$$Mv(x, m) = \begin{cases} 0, & x > \alpha_1 \text{ or } x < \alpha_2 & (a) \\ 1, & \beta_2 \leq x \leq \beta_1 & (b) \\ \frac{\alpha_1 - x}{\alpha_1 - \beta_1}, & \beta_1 < x \leq \alpha_1 & (c) \\ \frac{x - \alpha_2}{\beta_2 - \alpha_2}, & \alpha_2 \leq x < \beta_2 & (d) \end{cases} \quad (1)$$

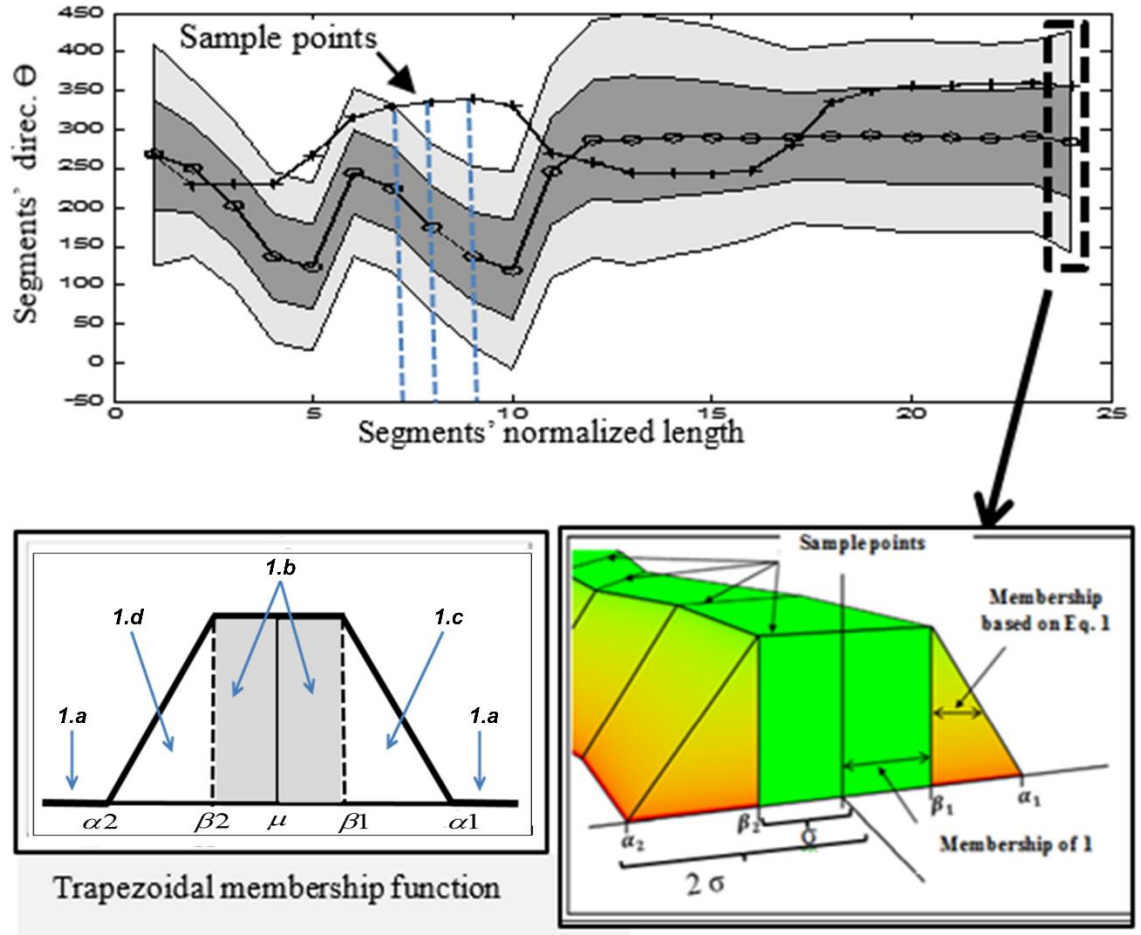


Figure 3-12 Fuzzy comparison between a sample and a model

The values of α_1 , α_2 , β_1 , and β_2 were taken as fixed values in [82] and [83] while here these values are automatically generated from the training samples.

The membership value (M_v) represents the weight of the similarity between sample x and the model at sampling point m . The value of $M_v \in [0, 1]$ such that '0' (1.a) means least similarity between the sample (outside the shaded region) and the model at that sampling point. Membership value '1' (1.b) represents the maximum similarity (central region). Membership value in the light shaded area (outer region) is calculated using (1.c) and (1.d) of Equation 1. Figure 3-12 shows these four cases for digit sample S and a digit model M , N is the number of points. The overall similarity between M and S is given by

$$\text{sim}(S, M) = \frac{1}{N} \sum_{i=1}^N M_v (S(i), M(i))$$

The zoomed part in Figure 3-12 illustrates the fuzzy model in 3-dimentional representation. This illustration shows the real form of the fuzzy model, including the two tolerance regions. The figure shows a number of sampling points of the model; the similarity check is done at each one of these points.

To enhance the achieved recognition rates, we used histogram-based features to validate the decision taken by the fuzzy classifier as follows. Two sets of features are extracted from the testing sample, directional and histogram-based features. The first one is used by the fuzzy classifier to identify to which class the sample belongs to. The second one is used to validate the recognized class. The decision taken by the fuzzy classifier is evaluated based on the histogram-based features of the sample and the selected class. If they are similar,

then the classification confidence of this decision is accepted. Otherwise, the selected class is rejected and the control goes back to the fuzzy classifier to select the next candidate class. The validation process is illustrated in figure 3-13.

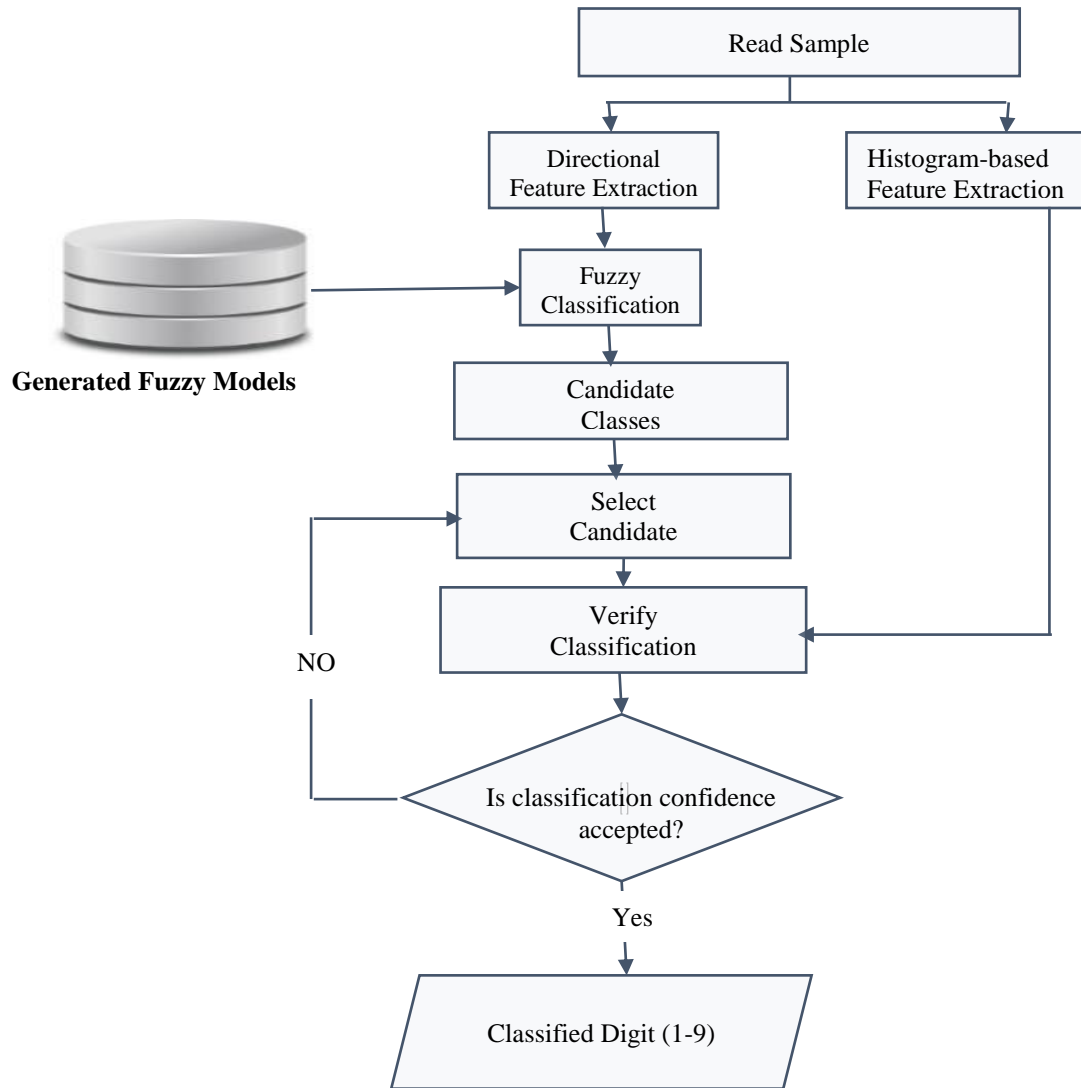


Figure 3-13 Fuzzy classification block diagram

3.6 Experimental Work

In this section we present the experiments that were conducted to evaluate our technique for the recognition of Arabic online digits. In addition, the details of the used database and the results are discussed.

In our experiments, we used the Arabic On-line Digits Database (AOD*) proposed by Abdul Azeem et al. [14]. AOD was collected from 300 writers of varying ages and without enforcing any constraints on digit size, orientation or number of strokes per digit. More than 32,000 online Arabic digits were collected by asking each writer to write an average of 10 samples per digits. About 78% of the AOD database was used for training and the remaining 22% for testing. The proposed technique is composed of two stages. In the first stage, all testing data are classified into zero or nonzero digits. SVM classifier with RBF was used in this stage. The classifier was designed for two-class problem with five dimensional feature vector (length, width, height, height variance and aspect ratio). An overall accuracy of 99.55% was achieved in this phase.

The nonzero digits are classified by using the fuzzy classifier in the second stage. A 20% of the training data is selected as validation data which is used for fuzzy model parameters' estimation. The values for the first tolerance (β_1 and β_2) and the second tolerance (α_1 and α_2) are chosen experimentally as $(\sigma/4)$ and $(\sigma/2)$, respectively. These parameters are used in the experiments with the extracted features of the test data. A recognition rate of 93.36%

* Available at: \ <http://www.aucegypt.edu/sse/eeng/Pages/AOD.aspx>

was obtained in this stage using the first candidate. To improve the recognition rate of this stage, we use histogram-based features to validate the decision taken by the fuzzy classifier as discussed in section 3.4 such that if the decision based on the first candidate is inadequate the next candidate of the fuzzy classification is selected. After applying this improvement, the recognition rate reached 98%.

We evaluated our directional features using SVM classifier to classify Arabic digits. An average recognition rate of 93% is achieved. This indicates that our fuzzy models with the proposed fuzzy structural classifier are more effective than using the directional features with SVM classifier for online Arabic digit recognition. Table 3-1 shows the confusion matrix of our classification which includes fuzzy-based and histogram-based classification.

Table 3-1 Confusion matrix of the enhanced fuzzy classification

| Digit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | R.R. |
|---------------------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|--------------|
| 1 | 836 | 2 | 4 | 0 | 0 | 0 | 1 | 3 | 1 | 98.7 |
| 2 | 1 | 653 | 2 | 7 | 0 | 1 | 3 | 0 | 1 | 97.8 |
| 3 | 5 | 1 | 651 | 1 | 0 | 0 | 4 | 6 | 6 | 96.6 |
| 4 | 0 | 8 | 0 | 676 | 0 | 3 | 1 | 1 | 1 | 97.9 |
| 5 | 0 | 2 | 0 | 0 | 694 | 0 | 0 | 0 | 2 | 99.4 |
| 6 | 11 | 0 | 0 | 3 | 0 | 650 | 2 | 0 | 14 | 95.6 |
| 7 | 1 | 0 | 0 | 0 | 0 | 0 | 677 | 0 | 0 | 99.9 |
| 8 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | 661 | 1 | 99.3 |
| 9 | 0 | 6 | 1 | 2 | 2 | 9 | 0 | 2 | 659 | 96.8 |
| Average Recognition Rate | | | | | | | | | | 98.01 |

The misclassified samples of the fuzzy classifier were analyzed carefully and it was found that some of these samples are badly written and may not be recognized by human. Therefore, subjective evaluation was conducted on the misclassified samples. The images of the misclassified samples were printed randomly in a form and the form is given to 30 graduate/undergraduate students. Each student was told that these images are images of Arabic digits and he was asked to label the images as he perceives in the image. The results of this subjective evaluation are summarized in Table 3-2.

Table 3-2 The overall classification results of the human subjective evaluation

| Digit | Correctly Classified | Incorrectly Classified | Undetermined |
|----------------|-----------------------------|-------------------------------|---------------------|
| 1 | 1.67% | 42.22% | 56.11% |
| 2 | 86.89% | 0.00% | 13.11% |
| 3 | 80.14% | 4.93% | 14.93% |
| 4 | 83.33% | 10.48% | 6.19% |
| 5 | 45.83% | 4.17% | 50.00% |
| 6 | 64.19% | 13.87% | 21.94% |
| 7 | 0% | 10% | 90% |
| 8 | 67.33% | 11.33% | 21.33% |
| 9 | 64.85% | 16.36% | 18.79% |
| Average | 64.77% | 13.12% | 22.11% |

As shown in the table, the labels of the students are classified into three groups. The first group includes the percentage of answers that correctly labeled the digits. The second group indicates the percentage of answers that labeled the digit incorrectly. The last group

includes the percentage of responses where the students were unable to label the images. The subjective evaluation of the results shows that most students were not able to classify digits ‘1’ and ‘7’ correctly. Digits 5, 6, 8 and 9 were correctly labeled by no more than 68% of the subjective evaluators. Some samples of digits 2, 3 and 4 were not reflecting the proper directional way of writing these digits. The order of writing of these samples is different from the normal order of writing similar samples as shown in figure 3-14. It is not easy to recognize these samples based on their directional features. However, the images of these samples can be recognized by humans easily.

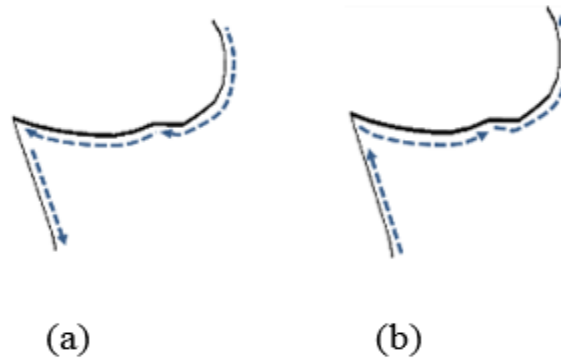








Figure 3-14 (a) Regular order of writing digit ‘three’ (b) Irregular order of writing digit ‘three’

The overall results show that $\approx 65\%$ of the samples were classified correctly by humans, $\approx 13\%$ were incorrectly classified and the remaining $\approx 23\%$ were undetermined. Therefore, $\approx 35\%$ of the misclassified digits by our fuzzy classifier were not classified by humans.

In our work, we are using the same database used by Abdul Azeem et al. [14]. However, they used offline features in their approach by converting the user’s strokes into a bitmap image while in our work we are using the online features. In addition, they used 30000 digits in their experiments whereas in our experiments we used all the samples of AOD

database (32695 digits). Hence, the results of both techniques may not be comparable. We could not use their 30000 samples as we have no information of which 30000 samples were used. In one experiment when we removed unreadable samples by humans (125 samples) our combined recognition rate (stages 1 and 2) reached 99.55%.

Table 3-3 Some misclassified samples of the fuzzy classifier

| Digit Image | Digit Class | Classified Class |
|--|-------------|------------------|
|  | ٣ | ٧ |
|  | ٤ | ٩ |
|  | ٥ | ٢ |
|  | ٦ | ٤ |
|  | ٨ | ٩ |
|  | ٩ | ٤ |

3.7 Conclusions

In this chapter, a technique based on automatic fuzzy modeling for the automatic recognition of Arabic (Indian) online digits is presented. In this technique we automatically generate fuzzy models of the different digits using the segments' directions of Arabic online digits of the training data. The models include the samples' trend line of each digit and the digit's model upper and lower envelopes. The fuzzy intervals are generated automatically based on the analysis of the training samples at the digit segment level and not set manually at the digit level as in the previous works. In addition, we automatically generate weights for the different segments using the training samples. These weights are integrated in the fuzzy similarity estimate.

For the classification, a two stage approach is implemented where SVM is used in the first stage (using statistical features) and fuzzy-based approach using the automatically generated models in the second stage. The second stage has an integrated feedback verification step which verifies the recognized test sample label. If the first label does not pass validation (using other features) then the next label in the list is selected in the feedback loop otherwise it is selected as the recognized digit.

A database containing more than 30,000 Arabic online handwritten digits is used to test the proposed approach. An overall accuracy of 99.55% was achieved in the first stage (zero/nonzero) and the second stage (digits 1 to 9) achieved an accuracy of 98.01%. This result, based on using our fuzzy models and the proposed fuzzy structural classifier, proved to be better than using the SVM classifier with the directional features. The misclassified samples are evaluated subjectively and results indicate that humans were able to recognize $\approx 65\%$ of these samples.

CHAPTER 4

ONLINE ARABIC CHARACTER RECOGNITION

BASED ON GRAPHEME MODELING

In this chapter, a grapheme-based approach for recognizing isolated online Arabic characters is presented. The novelty of this work comes from modeling the isolated online Arabic characters based on their graphemes using the generated graphemes' codebook.

This chapter is organized as follows. The introduction is presented in section 4.1 then the collected dataset is described in section 4.2. Section 4.3 presents the preprocessing steps followed by the graphemes' extraction process in section 4.4. Features used in this work are detailed in section 4.5; section 4.6 describes the proposed method for generating the graphemes' codebook. The experimental results are reported in section 4.7; and finally the conclusions are presented in section 4.8.

4.1 Introduction

Handwritten character recognition is an area of pattern recognition that has received considerable interest during the last decades. Most of the research on Arabic characters addressed offline handwritten characters [7] [8] while few addressed online Arabic characters [84].

In this chapter we propose a grapheme-based approach for the automatic recognition of Arabic online characters. The proposed technique is based on modeling the characters based on their graphemes. In the training phase, the characters are segmented into a set of small and basic parts called graphemes. Then, the segmented graphemes are clustered regardless of the characters to which these graphemes belong to. The graphemes' codebook is generated from the representative samples of all clusters. The codebook is used to represent the characters based on their graphemes. In the recognition phase, the codebook is used to recognize the testing characters based on their extracted graphemes. This is not a trivial task since deciding which grapheme belongs to which character is a challenging task and it needs an effective algorithm [6]. Different features and different classification approaches are used in order to investigate the proposed modeling. The proposed approach is evaluated using a dataset of isolated online Arabic characters collected from 20 writers and comprises the different forms of all Arabic characters.

Two sets of experiments are conducted in this work. The first set was designed to evaluate the extracted graphemes themselves regardless of the characters to which these graphemes belong. The aim of these experiments was to evaluate the extracted graphemes by recognizing the testing graphemes based on the models generated from the training graphemes. In the second set of experiments, the testing characters are recognized using the characters' graphemes modeling of the training characters. Figure 4-1 shows the overall block diagram of the proposed approach.

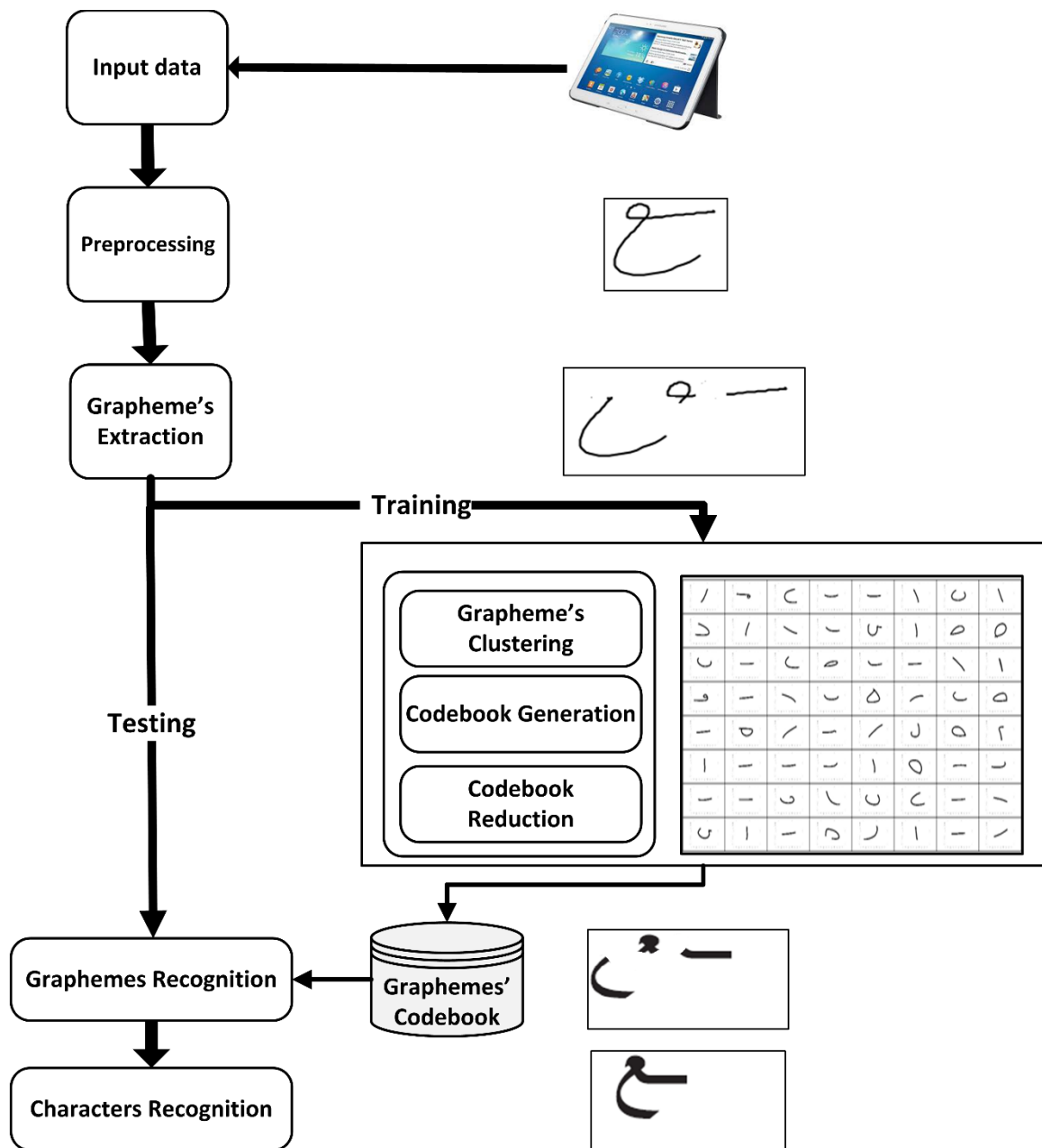


Figure 4-1 Overall block diagram of the proposed approach for Arabic online character recognition

4.2 Collected Data

To model the online Arabic characters, a dataset of isolated online Arabic characters have been collected from 20 writers. The dataset was collected using InkMIPad application, which is an ink editor application used to create ink files on windows platform. InkMIPad uses Tablet PC SDK to capture the digital ink and it saves the collected data as InkMI files [87]. Figure 4-2 shows the editor of InkMIPad application.

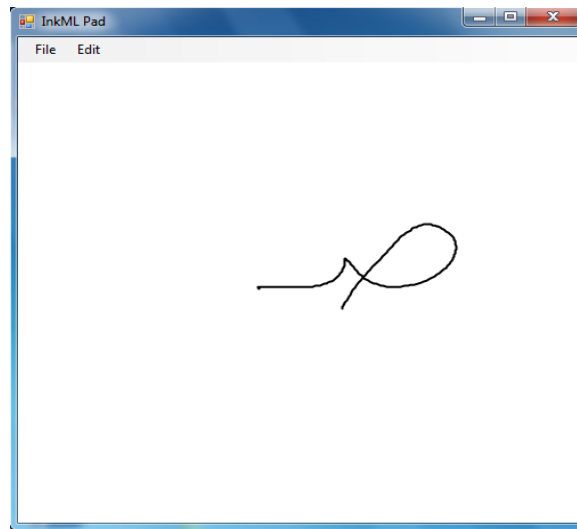


Figure 4-2 InkMIPad Editor

InkMIPad represents the drawn character using a sequence of points and each point is represented using its (x, y) coordinates. The collected dataset contains 2160 samples and covers the different shapes of all Arabic characters i.e. the beginning, middle, end and isolated forms. No constraints are enforced on the size, orientation or number of strokes per character. Figure 4-3 shows some samples from the collected characters.

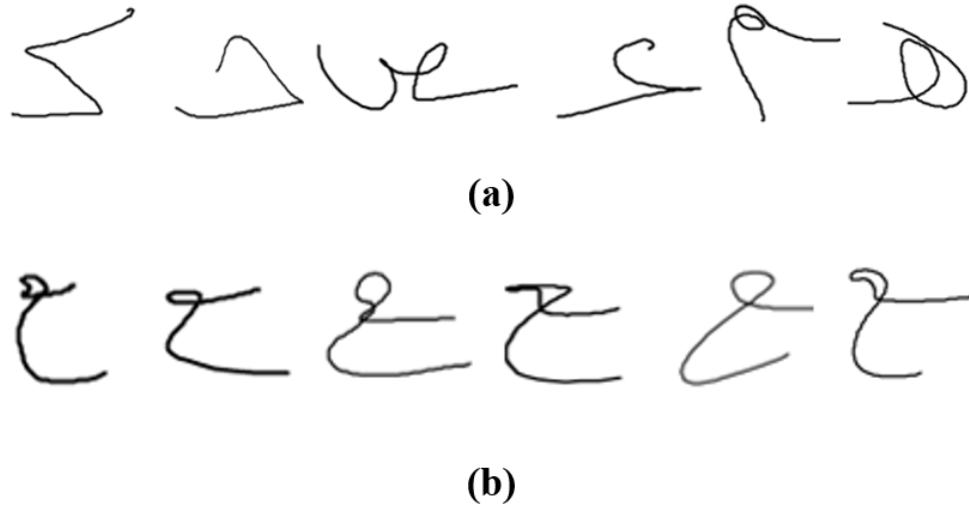


Figure 4-3 Samples from the collected dataset. (a) Samples of different characters (b) Different samples from the end form of Arabic letter “Ain”

4.3 Preprocessing steps

Preprocessing phase is important in order to achieve better accuracy. In our work, simplification, removing duplicated points, smoothing and length normalization are applied to enhance the collected data as follows. Douglas-Peucker algorithm [85] is used to simplify the character’s trajectory by remove unimportant points. The duplicated points are removed by checking if any two points are equal, if so one of them is removed. The character’s trajectory is also smoothed by moving a window along the grapheme’s trajectory points and replacing each point inside the window with the window’s mean value. Finally, the lengths of all samples are normalized by adjusting all samples’ trajectories to a unique length. Figure 4-4 shows the middle form of Arabic character ‘Ain’ before and after smoothing.

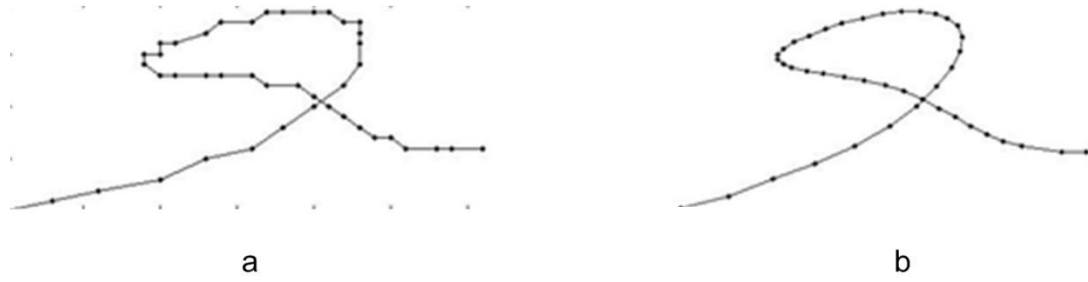


Figure 4-4 Preprocessing for the middle form of Arabic character ‘Ain’. (a) Before smoothing. (b) After smoothing

4.4 Graphemes extraction

In the literatures, different algorithms are proposed for decomposing characters into graphemes. Jung, S.K. Kim [88] extracting the graphemes based on selecting the corner points as the cut points. The chain code representation is used to detect the local maximum convex or concave curvatures of a stroke. In 2010, De Cao Tran [6] extracted the graphemes based on the maximum and the minimum values of y-coordinate. Thus, the extracted grapheme is a sequence of points from maximum y-coordinates to minimum y-coordinates or from minimum to maximum. A recent approach was used by Boubaker et al. in 2014 [89] and also in [90]. Their approach extracts the graphemes by detecting two points; bottom of the valleys and the angular points i.e. the extremum point where the curve turns back.

In this work, the proposed algorithm for extracting the graphemes is as follows.

1. Checking if the character has a loop: if there is a loop, the loop’s trajectory represents a grapheme.

2. For the non-loop points, measuring the curvature of the character's trajectory by detecting the sharp turn in the trajectory: if the curvature is greater than a threshold then that is a cut point.

Figure 4-5 shows the end form of letter 'Ain' before and after segmenting it into its graphemes.

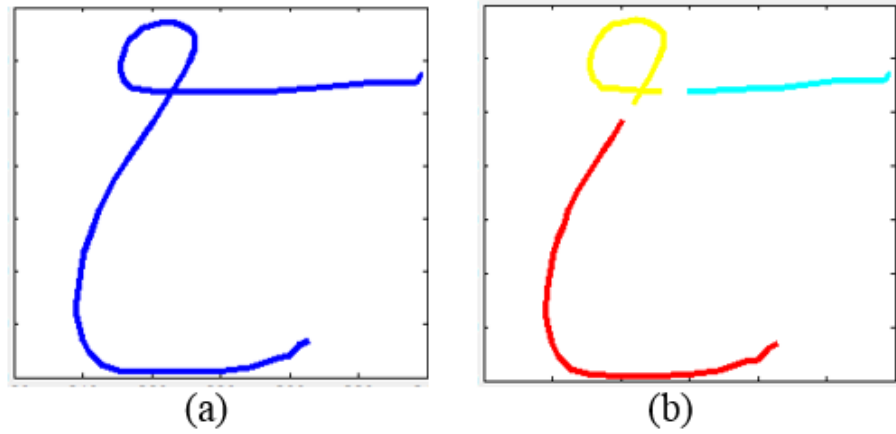


Figure 4-5 Grapheme's Extraction. (a) The end form of character 'Ain'. (b) The extracted graphemes in different colors

4.5 Feature Extraction

After extracting the graphemes, the following features are used to represent the graphemes:

4.5.1 Writing Direction Feature

The writing direction at any point $p(x(t), y(t))$ is calculated using sine and cosine functions [91] where:

$$\sin(\alpha t) = \frac{\delta y(t)}{\delta s(t)}, \text{ and } \cos(\alpha t) = \frac{\delta x(t)}{\delta s(t)}, \text{ where}$$

$$\begin{aligned}\delta x(t) &= x(t-1) - x(t), \\ \delta y(t) &= y(t-1) - y(t), \text{ and} \\ \delta s(t) &= \sqrt{\delta x^2(t) + \delta y^2(t)}\end{aligned}$$

For each grapheme, the writing direction features are extracted twice. The first one represents the writing directions of the whole grapheme points and the second one represents the writing directions of the grapheme's points in each window after partitioning the grapheme into four windows as shown in figure 4-6.

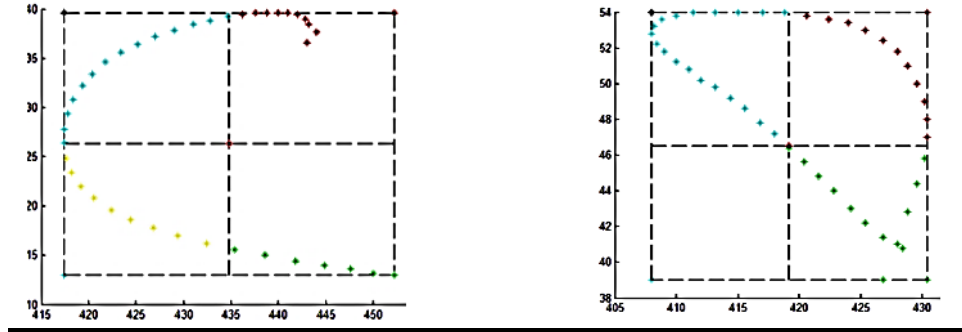


Figure 4-6 partitioning the grapheme's trajectory into four windows

4.5.2 Orientation Histogram-based features

This feature is based on the histogram of grapheme's segments orientation. The details of this feature is presented in Chapter 3 (section. 3.3.3)

4.5.3 Polar Angular features

These features are based on the relationship between the point in the center of the bounding box surrounding the grapheme and all grapheme's points. As a preprocessing step, the size normalization process is performed to obtain a uniform size for all graphemes [92]. As a

result of this normalization process, the grapheme's center of gravity will be the origin point (i.e. point (0, 0)). The normalized grapheme's trajectory is calculated as follows. Given the grapheme's trajectory $\mathbf{G} = \{\mathbf{P}_i\}_{i=1}^n = \{(x_i, y_i)\}$, the normalized grapheme's trajectory is $\bar{\mathbf{G}} = \{\bar{\mathbf{P}}_i\}_{i=1}^n = \{(\bar{x}_i, \bar{y}_i)\}$, such that:

$$\bar{x}_i = \frac{(x_i - \mu_x)}{W} \quad \text{and} \quad \bar{y}_i = \frac{(y_i - \mu_y)}{W}$$

Where, W is the maximum of (the width, the height) of the grapheme's trajectory; and

$$\mu_x = \frac{1}{N} \sum_{i=1}^N x_i \quad \text{and} \quad \mu_y = \frac{1}{N} \sum_{i=1}^N y_i$$

Figure 4-7 shows samples of different graphemes (red curves) with lines connecting the grapheme's points with the center of gravity. The polar angular feature is described by the lengths and directions of these lines (the blue lines).

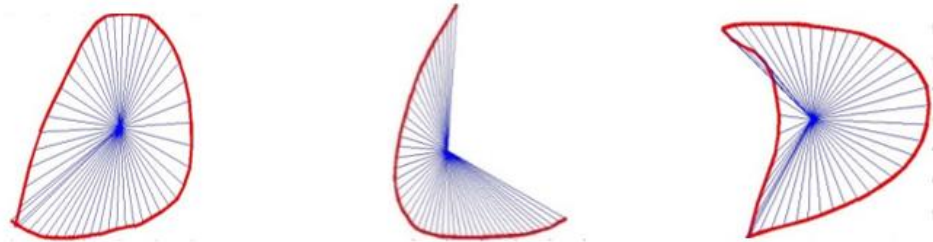


Figure 4-7 Samples of different graphemes (red curves) with lines connecting the grapheme's points with the center of gravity (blue lines)

4.5.4 Chain Code features

The freeman chain code representation [93] , which is a popular technique for representing images, is also used in our work to represent the grapheme's trajectory as follows.

Let $Seg_i = 1, 2, 3, \dots, n$, be the segments of a grapheme G . Each segment is represented by an integer $(0, 1, \dots, 7)$ based on its direction as shown in figure 4-8.

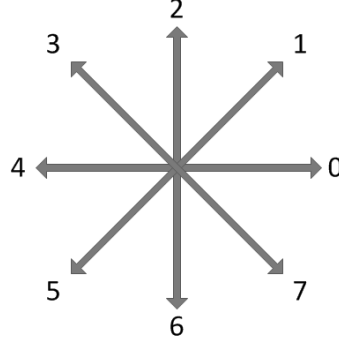


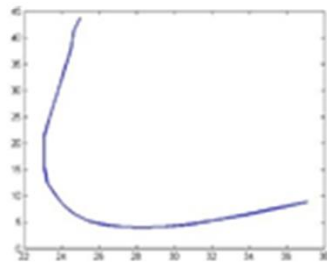
Figure 4-8 Freeman chain code

4.5.5 Curliness features

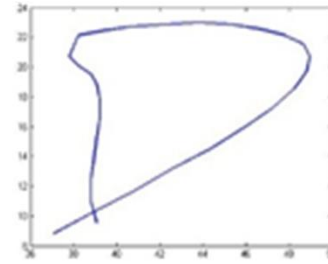
This feature describes the curving of the grapheme (i.e. the deviation of grapheme points from a straight line). The calculation of this feature is based on the length of the grapheme's trajectory and the maximum side of the bounding box [13] as follows

$$Curliness(G) = \frac{Length(G)}{\max(L_x, L_y)} - 2$$

Where, $Length(G)$ represents the length of the trajectory; L_x and L_y are the width and the height of the grapheme's trajectory. Figure 4-9 shows the curliness values of two different graphemes.



Curliness = -0.6813



Curliness = 0.9662

Figure 4-9 Curliness feature

4.5.6 Loop detection feature

This feature indicates whether the grapheme points form a loop or not. This is done by checking the self-intersection in the grapheme's trajectory. The values of this feature are Boolean expressed as true or false.

4.5.7 Other features

In addition to the previous features, the following features are used to classify the different graphemes' classes.

1. The grapheme's length.
2. The grapheme's width.
3. The graphemes' height.
4. The aspect ratio.
5. The horizontal direction (right to left or left to right).
6. The vertical direction (top to bottom or bottom to top).

4.6 Graphemes' Codebook Generation

This section describes the process of generating the graphemes' codebook from the extracted graphemes. The process starts by clustering the graphemes into different classes, then the representative samples of these classes are used to build the codebook. The inter-class similarities are used after that to reduce the size of the generated codebook. The details of this process is described in the following subsections.

4.6.1 Graphemes Clustering

The features presented in section 4.5 (i.e. writing direction, histogram-based orientation, polar Angular, chain code, curliness, existence of loop and other features) are extracted from all graphemes of the training characters, then *k-means* clustering algorithm is used to cluster the graphemes into different clusters based on the extracted features. The algorithm is run many times with different number of clusters and different combination of features. The generated clusters are evaluated subjectively with each run to get the best clusters. The best results were achieved by using 64 clusters.

4.6.2 Codebook Generation

The mean sample of each class is not a good representative for that class thus these mean samples are not the best choice for representing the classes in the codebook. To generate the codebook, the intra-class similarities are used to select the representative sample for each class as follows. The distance between each sample and all other samples, within the same class, are calculated, and then the sample with minimum distance with all samples is

selected as a representative for the class. The codebook is generated from the representative samples of all classes.

4.6.3 Codebook Reduction

In order to reduce the size of the codebook, inter-class similarities are used to merge similar classes into the same bag-of-classes. Each bag-of-classes contains a class or a set of classes that are similar. The distances are calculated and the classes with distance less than a threshold are merged into the same bag. Our experiments, as described in section 4.7, show the effectiveness of reducing the number of classes using our implemented bag-of-classes.

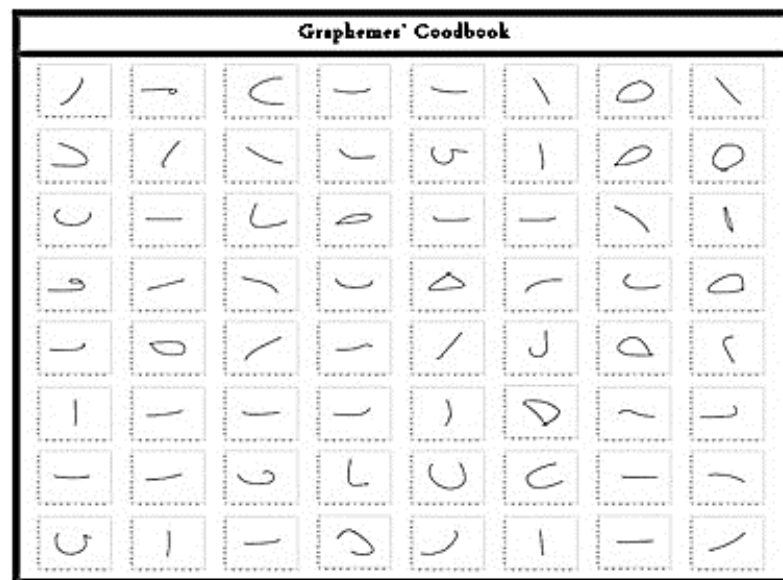


Figure 4-10 Graphemes' Codebook

4.7 Experimental work

In this work, two sets of experiments were conducted. The first set was designed to evaluate the extracted graphemes regardless to which characters they belong. The generated classes and the reduction of the original 64-classes into 34-bag-of-classes are evaluated in this set of experiments. In the second set, the testing characters are recognized using the characters' graphemes modeling of the training data. The following subsections present the details of our experiments.

4.7.1 Grapheme Classification

In these experiments, the models generated from the training graphemes are used to recognize the testing graphemes. The generated classes and the generated codebook were evaluated in these experiments as described in the following subsections.

4.7.1.1 Evaluating the generated classes

In the first experiment we find the representative sample of each Bag-of-classes; then we generate the fuzzy models of these representative samples. The membership values of all graphemes with the generated fuzzy models are calculated. Then, each grapheme is represented by the following two features:

- a) The segments' directions of the grapheme.
- b) The membership values with the closest fuzzy models.

SVM classifier is used to recognize the testing graphemes based on the training graphemes and the obtained recognition rate was 91.66. However, the recognition rate was improved

to 95.17 after combining the features (a) and (b) with the graphemes-features presented in section 4.5 (i.e. writing direction, histogram-based orientation, polar Angular, chain code, curliness, existence of loop and other features).

We modify the setting of the previous experiment by representing the segments' direction using its standard writing direction number instead of the angle. Standard writing direction divides the 360^0 degree into sixteen directions i.e. each direction covers 45^0 with overlap of 22.5^0 between each two consecutive regions as shown in Figure 3-6(a). Then, the following features are extracted for all data:

- a) Standard writing direction numbers (two numbers representing the two regions since we are allowing the overlap).
- b) The membership values (two values).

A recognition rate of 90.20 was achieved using these features. The following table summarizes the experiments conducted to evaluate the generated classes.

Table 4-1 The results of evaluating the generated classes

| Experiment 's description | Classifier | Recognition Rate |
|---|------------|------------------|
| Standard writing direction numbers and the membership values are used to represent all graphemes. | SVM | 90.20 |
| The grapheme's segments' directions (angles) and the membership values are used to represent all graphemes. | SVM | 91.66 |
| Combining the grapheme's segments' directions and the membership values with the following features: writing direction, orientation histogram-based, polar Angular, chain code, curliness and existence of loop features. | SVM | 95.17 |

4.7.1.2 Evaluating the reduction of the codebook using Bag-of-classes

These experiments were designed to evaluate the reduction of the original 64-classes into 34-bag-of-classes. Two experiments were conducted where in the first one the original generated codebook (before reduction) was used to label the graphemes. Therefore, graphemes are labeled here based on the closest class from the 64-classes. Then we use fuzzy classification (described in section 3.5.2) to recognize the testing graphemes using the fuzzy models generated from the training graphemes; here the fuzzy models are generated for each Class. A recognition rate of 77.18 was achieved in this experiment. This recognition rate was improved to reach 90.11 after adjusting the labeling of all training and testing graphemes to be based on the closest Bag from the 34-bags (after the reduction). The fuzzy models here are generated for each Bag. This improvement proves the feasibility of reducing the original classes using Bag-of-Classes.

The following table summarizes the experiments of evaluating the reduction of the codebook using bag-of-classes along with the achieved recognition rates.

Table 4-2 The reduction of the codebook using bag-of-classes

| Experiment's Description | Classifier | Recognition Rate |
|--|----------------------|------------------|
| - Before Reduction: Using the original generated 64-Classes (<i>i.e.</i> The fuzzy models are generated for each Class). | Fuzzy classification | 77.18 |
| - After Reduction: Using the 34- Bag-Of-Classes (<i>i.e.</i> The fuzzy models are generated for each Bag). | Fuzzy classification | 90.11 |

To illustrate how the codebook size affects the recognition rate, we conducted another experiment with different codebook sizes. In this experiment, we represent all graphemes using the features presented in section 4.5 (i.e. writing direction, orientation histogram-based, polar Angular, chain code, curliness, existence of loop and other features) then we use different codebooks with different sizes as follows:

- a) The size of the codebook is **64**: The graphemes are labeled based on the closest class from the 64-Classes (before the reduction).
- b) The size of the codebook is **44**: The graphemes are labeled based on the closest bag (after reducing the classes into 44 bags).
- c) The size of the codebook is **34**: The graphemes are labeled based on the closest bag (after reducing the classes into 34 bags).

For each case, we recognize the testing samples based on the training samples using SVM and the following recognition rates were achieved 93.27, 94.23 and 97.02 for codebook sizes 64, 44 and 34 respectively. The highest recognition rate (97.02) was achieved after reducing the original classes to 34 bags and this also shows the effectiveness of reducing the size of the codebook. The following table shows the impact of codebook size reduction on recognition rate

Table 4-3 The impact of codebook size on recognition rate

| Experiment's Description | Classifier | Recognition Rate |
|---|------------|------------------|
| Codebook size= 64 (<i>Before reduction</i>) | SVM | 93.266 |
| Codebook size= 44 (<i>reducing the classes to 44 bags</i>). | SVM | 94.219 |
| Codebook size= 34 (<i>reducing the classes to 34 bags</i>). | SVM | 97.02 |

4.7.2 Character Classification

In these experiments, the testing characters are recognized using the characters' graphemes modeling of the training data. In the graphemes' extraction phase, the characters are segmented into small and basic parts called graphemes. Then in the recognition phase, the representations of these graphemes are combined to build the pattern of the corresponding character. Different representations are proposed in order to achieve better recognition rate.

Initially, we represent each character by a 34-length vector representing the 34 *Bag-of-classes*. The values of this vector are 0's and 1's such that the Bag number is set to one if the character has a grapheme from that bag or zero otherwise. A recognition rate of 60.34 was achieved using this representation. This representation suffers from generating sparse vectors i.e. vectors that have mostly zero values. To overcome the problem of sparse vectors we use the similarities between the character's graphemes and the *Bag-of-classes* instead of using binary representation. The recognition rate reaches 77.469 after using this representation which is an improvement over the binary representation but still need to be improved.

These experiments show that bag-of-classes alone is not enough for representing the characters based on their graphemes. More features may be needed to improve the accuracy. In this regard, a set of features are proposed to represent the characters. The proposed features are described in the next section.

4.7.2.1 Characters' representation

We propose to use the following features to represent the characters based on their graphemes:

a) The grapheme's Bag

The value for this feature will be a number representing the *Bag* number to which the grapheme belong (i.e. a number between 1 and 34).

b) The ratio between the grapheme's length and the character's lengths

The relative length feature of the grapheme is important in distinguishing between the graphemes that look similar but have different length. The value for this feature will be greater than ZERO and less than or equal to ONE.

c) The grapheme's location

The location is represented by four values and each value represents the percentage of the grapheme in one window (the character is divided into 4-windows). Thus this feature has four values:

- C1: The percentage of the grapheme in the Top-Right window.
- C2: The percentage of the grapheme in the Bottom-Right window.
- C3: The percentage of the grapheme in the Top-Left window.
- C4: The percentage of the grapheme in the Bottom-Left window.

Figure 4-11 shows the percentage of the grapheme of letter RAA in each window. Here the letter comprises only one grapheme and the values of this feature are 0.55, 0.25, 0 and 0.20 for C_1 , C_2 , C_3 and C_4 respectively.

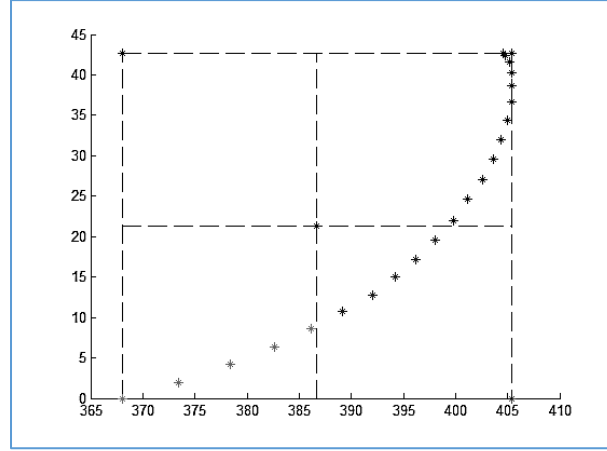


Figure 4-11 Grapheme' location

d) *Strat-End-Points feature*: this feature is related to the line connecting the start-point of the grapheme with the end-point of the grapheme (for example, the line connecting points S and E in Figure 4-12). Two features are extracted from this line which are:

d_1 : The length of the line connecting start-point with end-point.

d_2 : The direction of the line connecting start-point with end-point.

e) *Middle-Points feature*: this feature is related to the line connecting the middle-point of the line connecting the start-point of the grapheme with the end-point of the grapheme (for example, point M1 in Figure 4-12) and the middle point of the grapheme (for example, point M2 in Figure 4-12). Two features are extracted from this line which are:

e_1 : The length of the line connecting middle-points.

e_2 : The direction of the line connecting middle-points.

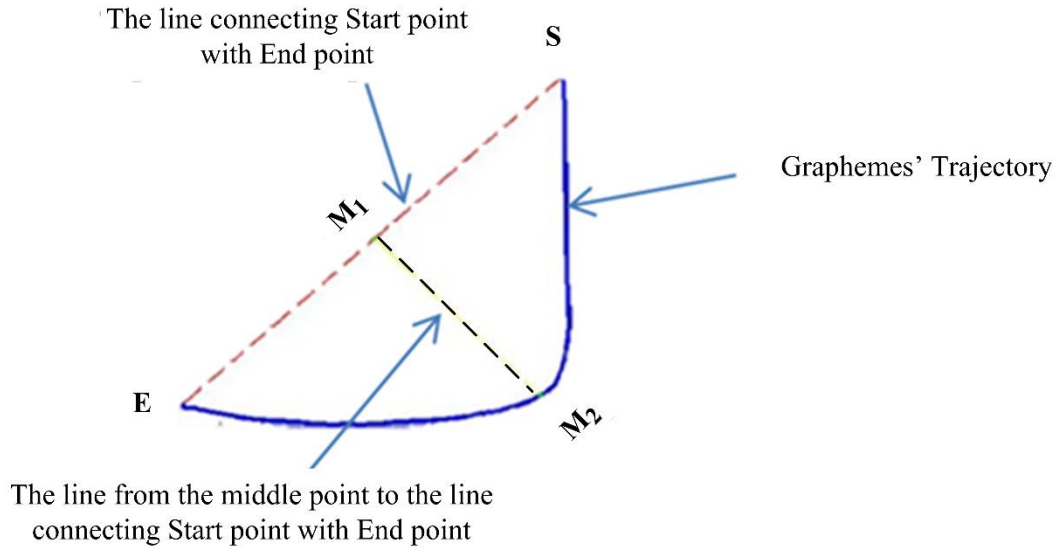


Figure 4-12 Sample of grapheme (the blue curve) where S represents the start-point, E represents the end-point, M_1 is middle-point of the line connecting the start-point with the end-point and M_2 is middle point of the grapheme

4.7.2.2 Used Classifiers

The following classifiers are used to classify the testing characters based on the models created from the training dataset:

- Support Vector Machine (SVM).
- Random Forest Tree.
- Logistic Model Tree (LMT).
- Multilayer Perceptron (MLP).

4.7.2.3 Experimental Results

Different experiments were conducted with different settings, the descriptions and results of these experiments are as follows.

In the first experiment, the features described in section 4.7.2.1 (i.e. the Bag No., the length ratio, the graphemes' location, the direction and length of the line connecting start point to end point and the direction and length of the line from the middle point of the grapheme to the line connecting start point to end point) were used to represent all characters. The classifiers presented in section 4.7.2.2 (i.e. SVM, Random Forest Tree, LMT and MLP) were used to classify the testing characters. The achieved character recognition rate in this experiment ranges from %78.53 to %86.91.

The same features were used in the second experiment after modifying features d_1 and e_1 by using the ratio of the length, with the grapheme length, instead of using the length itself. The best recognition rate obtained in this experiment is 87.53.

In the third experiment, the same features described were used after modifying features d_2 and e_2 as follows. Instead of using the angles for the direction, the numbers representing the standard writing directions are used. The number of directions used here are eight. The obtained recognition rate reaches 86.5743 in this experiment.

The modifications made in the second and third experiments for features d_1 , d_2 , e_1 and e_2 are combined in the fourth experiment, and the achieved recognition rate ranges from %78.83 to %86.67.

The same setting is also applied in the last experiment but the standard writing directions have different representation as follows. The number of directions will be 16 and there is an overlap between each two consecutive directions. The best recognition rate achieved in this experiment is %86.48. Table 4-4 summarizes the results of these experiments.

Table 4-4 The Recognition Rates of Online Arabic Characters based on Graphemes Modeling

| Experiment# | Description | SVM | Random Forest Tree | LMT | MLP |
|-------------|---|--------|--------------------|--------|--------|
| 1 | Using Direction and Length | 82.54 | 86.909 | 80.172 | 78.548 |
| 2 | Using Direction and length ratios | 84.286 | 87.530 | 79.217 | 78.930 |
| 3 | Using Standard Directions (8 directions) and Length | 80.952 | 86.575 | 79.742 | 78.691 |
| 4 | Using Standard Directions (8 directions) and length ratio | 82.882 | 86.670 | 79.647 | 78.835 |
| 5 | Using Standard Directions (16 directions with overlap) and length ratio | 77.619 | 86.479 | 78.930 | 80.077 |

As shown in the table, the best recognition rate is 87.5 and it was obtained using Random Forest Tree classifier. This result was achieved by using the following features: the Bag No., the length ratio, the graphemes' location, the direction and length ratio of the line connecting start point to end point and the direction and length ratio of the line from the middle point of the grapheme to the line connecting start point to end point.

Our experiments have been conducted using a self-collected data of isolated Arabic online characters. Table 4-5 shows the obtained recognition rates with the datasets used in techniques proposed for isolated Arabic online characters. The recognition rates of the different techniques are not comparable due to the different data used. However, we are showing them as a general reference.

Table 4-5 Recognition rates with the datasets used in systems proposed for isolated Arabic online characters

| Authors and Year | Dataset used | Reported Recognition Rate |
|--------------------------------|---|----------------------------------|
| Al-taani and Al-haj, 2010 [21] | 1400 Self-collected isolated character (Written by 10 writers). | 75% |
| Alddakiri and Bahaj, 2012 [94] | 1400 Self-collected isolated character (Written by 10 writers) | 83% |
| Ismail and Abdullah, 2012 [27] | 840 Self-collected isolated character, 60% for training. | 97% |
| Alijla and Kwaik, 2012 [95] | 585 Self-collected isolated character, 68% for training | 95.7% |
| Kour and Saabne, 2012 [52] | 5602 characters segmented from ADAB database | 91% |
| Proposed | 2160 Self-collected isolated character, 66% for training. (Written by 20 writers) | 87.5 |

4.7.3 Conclusions

In this work, an approach for recognizing isolated online Arabic characters based on graphemes' modeling is presented. The novelty of this work comes from modeling the isolated online Arabic characters based on their graphemes using the graphemes' codebook. In the training phase, the characters are segmented into a set of small and basic parts called graphemes. The codebook is generated from the extracted graphemes of the training characters. Then, the characters are represented based on their graphemes using the generated codebook. In the recognition phase, the graphemes of the testing characters are extracted then the representations of these graphemes are combined to build the pattern of the corresponding character. Different features are proposed and different classification

approaches are used in order to investigate the proposed graphemes modeling. A dataset of isolated online Arabic characters is collected to evaluate the proposed approach. The collected dataset comprises the different forms of all characters. Two sets of experiments are conducted in this work. The first set was designed to evaluate the graphemes classification. The aim of these experiments was to evaluate the extracted graphemes by recognizing the testing graphemes based on the models generated from the training graphemes. The achieved accuracy in these experiments reaches 97.02 which shows the effectiveness of the proposed process used in extracting the graphemes and building the codebook. In the second set of experiments, the testing characters are recognized using the characters' graphemes modeling of the training characters. A recognition rate of 87.53 was obtained in these experiments.

CHAPTER 5

ARABIC ONLINE TEXT RECOGNITION

This chapter describes the different phases of the proposed Arabic online text recognition. The proposed approach is based on segmenting the text into graphemes. In general, the text is composed of a set of graphemes and each one can represent a whole character or part of a characters' trajectory. In our approach, these graphemes are extracted and used in modeling the text. Different structural features are proposed to represent the extracted graphemes. For the classification, we use fuzzy classification to find the most similar class for each grapheme based on the automatically generated fuzzy models in the training phase whereas graph-based classification is used to recognize the graphemes of each character based on statistics gathered from the training characters.

The organization of this chapter is as follows. Section 5.1 presents an overview of Arabic Online Text Recognition process. The different steps of the training phase are described in section 5.2. Section 5.3 presents the details of the recognition phase. Arabic Online text recognition are detailed in section 5.4. The experimental results are discussed in section 5.5; and finally the conclusions are presented in section 5.6.

5.1 Overview of the Arabic Online Text Recognition Process

The framework for our proposed online text recognition is presented in figure 5.1. The framework consists of two main phases, the training phase and the recognition phase. The

training phase comprises two stages: codebook generation and fuzzy modeling. Although this framework is proposed for online text recognition, pre-segmented Arabic online characters are used in the training phase to generate the codebook and build the fuzzy models. Moreover, these segmented characters can be independent from the testing data which makes our approach more general. Unlike statistical classifiers that require large training data, our structural classifier can be trained using less data.

In the recognition phase, the Arabic online text is modeled as a sequence of graphemes which represent its basic parts. These graphemes are recognized based on the graphemes' fuzzy models generated in the training phase. Then, the recognized graphemes are mapped to their corresponding characters based on graphemes' statistics gathered by analyzing the training data.

5.2 Training phase

The grapheme models are built from online characters segmented from Arabic online text database (*Online-KHATT*). We choose to use a database of pre-segmented characters in the training phase instead of using isolated characters to better handle the issues related to characters' connections in an input text. Figure 5.2 shows a line from *Online-KHATT* database with some segmented characters.

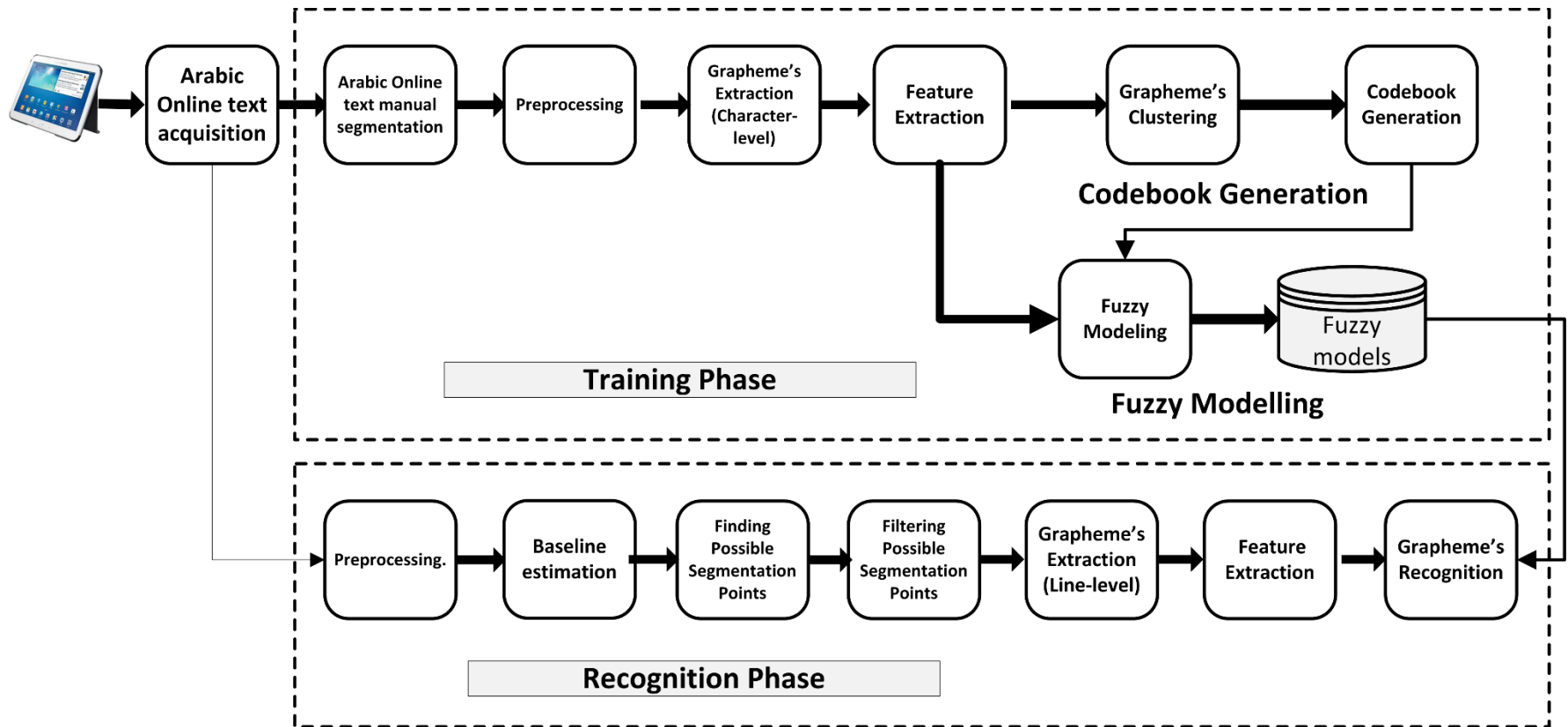


Figure 5-1 General framework for the proposed Arabic online text recognition

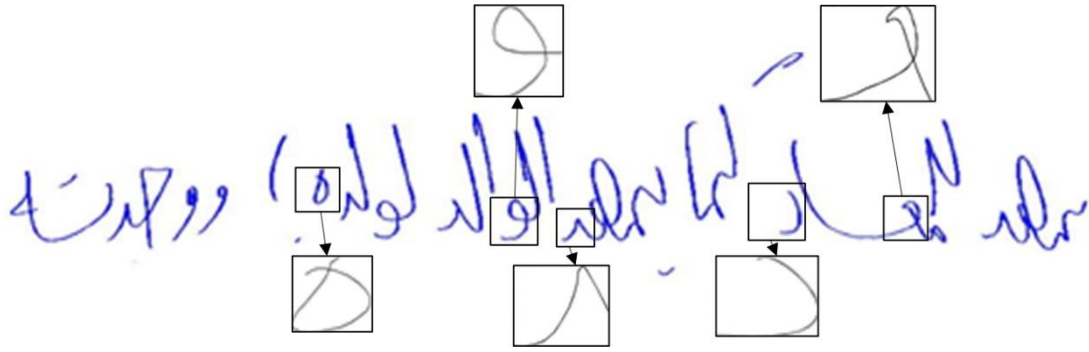


Figure 5-2 Arabic Online text from Online KHATT database with some segmented characters

In our work, the graphemes are extracted from the main stroke of the character, therefore the characters are grouped into classes based on the similarity in the main stroke. Table 5.1 shows the classes of Arabic characters based on common main stroke.

5.2.1 Preprocessing

The trajectories of the selected characters are enhanced by applying the following preprocessing steps: removing the repeated points, simplification using Douglas-Peucker algorithm [85] and smoothing. Figure 5-3 shows letter FAA before and after the simplification. The simplification's threshold is selected experimentally as 0.3 to avoid removing important points from the trajectory which will be the case if we increase the value of the threshold. The original trajectory is almost not simplified with values for the threshold smaller than 0.3. The details of these preprocessing steps are presented before in chapter 4 (section 4.3). In this work we also normalize the data by adjusting all the graphemes' lengths to 20 points. This length is derived based on the common lengths of samples. This is different than the one used with digits where the common lengths of the digits is 25 as shown in chapter 3.

Table 5-1 Characters' classes based on similarity in the main stroke

| Class | Label | Char | Class | Label | Char | Class | Label | Char |
|-------|----------|------|-------|----------|------|-------|----------|------|
| 1 | 'B_al' | ا | 19 | 'E_ya' | ي | 40 | 'I_laaa' | لا |
| | 'B_ba' | ب | 20 | 'E_fa' | ف | | 'I_laae' | لأ |
| | 'B_na' | ن | | 'E_ka' | ق | | 'I_laah' | لا |
| | 'B_ta' | ت | 21 | 'E_ha' | ح | 41 | 'I_lae' | لى |
| | 'B_th' | ث | | 'E_ja' | ج | 42 | 'I_ma' | م |
| | 'B_ya' | ي | | 'E_kh' | خ | 43 | 'I_na' | ن |
| 2 | 'B_ay' | ع | 22 | 'E_he' | ه | 44 | 'I_ra' | ر |
| | 'B_gh' | غ | | 'E_tee' | ة | | 'I_za' | ز |
| 3 | 'B_de' | د | 23 | 'E_laaa' | لا | 45 | 'I_se' | س |
| | 'B_sa' | ص | 24 | 'E_ma' | م | 46 | 'I_wa' | و |
| 4 | 'B_fa' | ف | 25 | 'E_na' | ن | | 'I_wl' | و |
| | 'B_ka' | ك | 26 | 'E_ra' | ر | 47 | 'E_ke' | ك |
| 5 | 'B_ha' | ح | | 'E_za' | ز | 48 | 'M_ay' | ع |
| | 'B_ja' | ج | 27 | 'E_se' | س | | 'M_gh' | غ |
| | 'B_kh' | خ | | 'E_sh' | ش | 49 | 'M_al' | ا |
| 6 | 'B_he' | ه | 28 | 'E_to' | ط | | 'M_ba' | ب |
| 7 | 'B_ke' | ك | 29 | 'I_aa' | ا | | 'M_na' | ن |
| 8 | 'B_la' | ل | | 'I_ae' | أ | | 'M_ta' | ت |
| 9 | 'B_laha' | لح | | 'I_ah' | إ | | 'M_th' | ث |
| 10 | 'B_lama' | لم | | 'I_am' | أ | | 'M_ya' | ي |
| 11 | 'B_ma' | م | 30 | 'I_ay' | ع | 50 | 'M_de' | د |
| 12 | 'B_se' | س | 31 | 'I_ba' | ب | | 'M_sa' | ص |
| | 'B_sh' | ش | | 'I_ta' | ت | 51 | 'M_fa' | ف |
| 13 | 'B_to' | ط | | 'I_th' | ث | | 'M_ka' | ك |
| | 'B_zha' | ظ | 32 | 'I_da' | د | 52 | 'M_ha' | ح |
| 14 | 'E_aa' | ا | | 'I_dh' | ذ | | 'M_ja' | ج |
| | 'E_ae' | أ | 33 | 'I_de' | ض | | 'M_kh' | خ |
| | 'E_ah' | إ | 34 | 'I_ee' | ي | 53 | 'M_he' | ه |
| 15 | 'E_ay' | ع | 34 | 'I_ya' | ي | 54 | 'M_ke' | ك |
| | 'E_gh' | غ | 35 | 'I_fa' | ف | 55 | 'M_la' | ل |
| 16 | 'E_ba' | ب | 35 | 'I_ka' | ق | 56 | 'M_ma' | م |
| | 'E_ta' | ت | 36 | 'I_ha' | ح | 57 | 'M_se' | س |
| | 'E_th' | ث | | 'I_ja' | ج | | 'M_sh' | ش |
| 17 | 'E_da' | د | 37 | 'I_he' | ه | 58 | 'M_to' | ط |
| | 'E_dh' | ذ | | 'I_tee' | ة | | 'M_zha' | ظ |
| 18 | 'E_de' | د | 38 | 'I_hh' | ه | 59 | 'E_la' | ل |
| | 'E_sa' | ص | 39 | 'I_ke' | ك | 60 | 'E_wa' | و |
| 19 | 'E_ee' | ي | | 'I_la' | ل | | 'E_wl' | و |

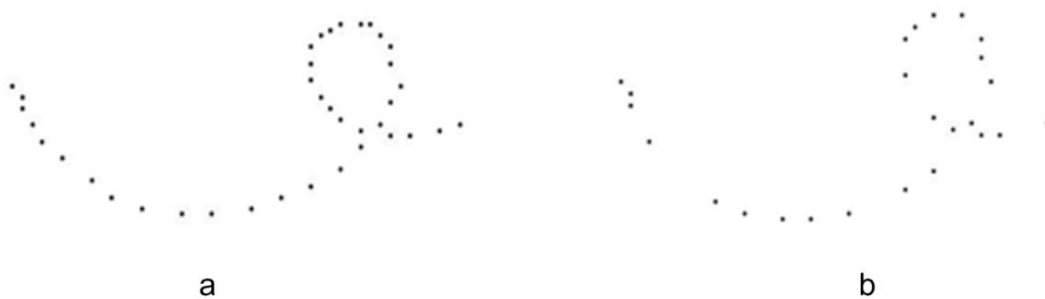


Figure 5-3 Letter FAA (a) before the simplification (b) after the simplification

5.2.2 Grapheme Extraction

The algorithm of extracting the graphemes from the segmented characters starts by extracting the loop from the character since the loop can be considered as a basic grapheme in modeling the character. Then, the sharp points are detected by measuring the curvature of the character's trajectory. These sharp points represent the cut points that are used to segment the character into its graphemes. Figure 5-4 shows some segmented characters along with their extracted graphemes.

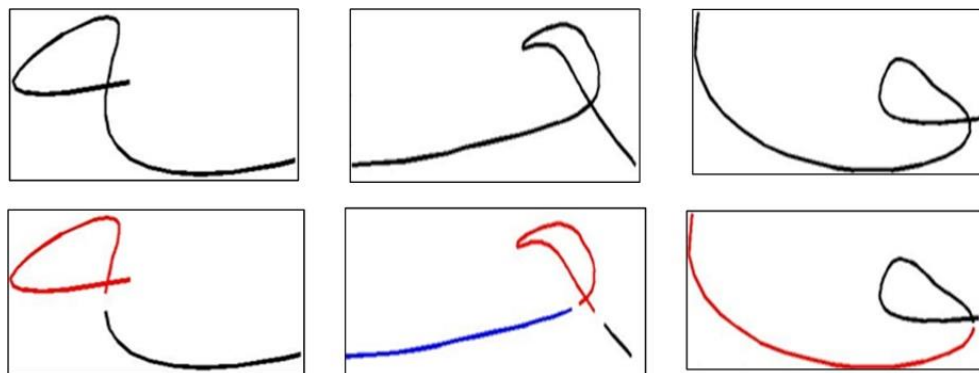


Figure 5-4 Some Arabic Online characters with their segmented graphemes

The sharp turn in the trajectory are detected as follows. Given three points $P1(x_1, y_1)$, $P2(x_2, y_2)$, and $P3(x_3, y_3)$, the curvature of a circle drawn through them is four times the area

of the triangle formed by these points divided by the product of its three sides [96]. The following equation and figure show the curvature of point P₁, P₂ and P₃.

$$\text{Curvature}(P_1, P_2, P_3) = \frac{4 * \text{Area of the triangle } (P_1, P_2, P_3)}{d(P_1, P_2) * d(P_2, P_3) * d(P_3, P_1)}$$

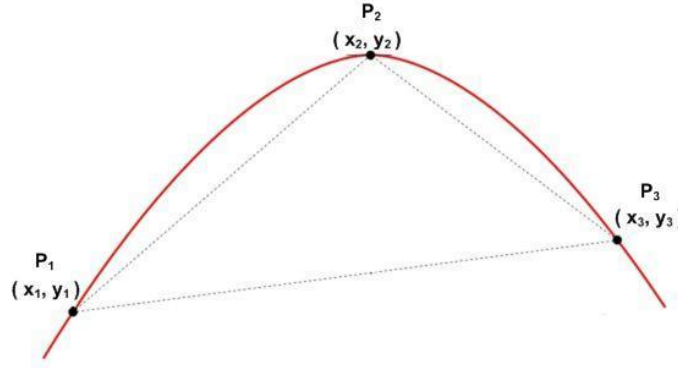


Figure 5-5 Curvature of Points P₁, P₂ and P₃

Figure 5-6 shows an example of Arabic PAW after applying the sharp turn formula, the sharp turns are marked using small red circles around them.

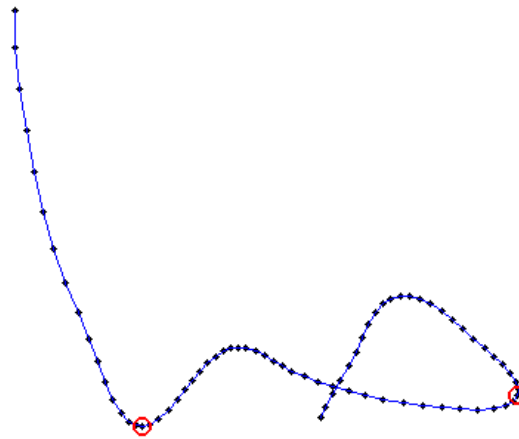


Figure 5-6 The sharp turns of Arabic PAW “صا” (marked by small red circles)

5.2.3 Feature Extraction

This section presents the features used to represent the extracted graphemes. Some features have been presented in details in the previous chapters. Therefore, these features will be listed with references to the details in the previous chapters.

5.2.3.1 Angles Quantization Feature

The grapheme is represented by the angles of its segments and the regions of these angles where the standard directions are divided into 8 regions of 45° . The Angles Quantization feature vector $AngQuant = [a_1, r_1, a_2, r_2, \dots, a_{n-1}, r_{n-1}]$, where a_i is the angle between point P_i and point P_{i+1} and r_i is the region's number of angle a_i .

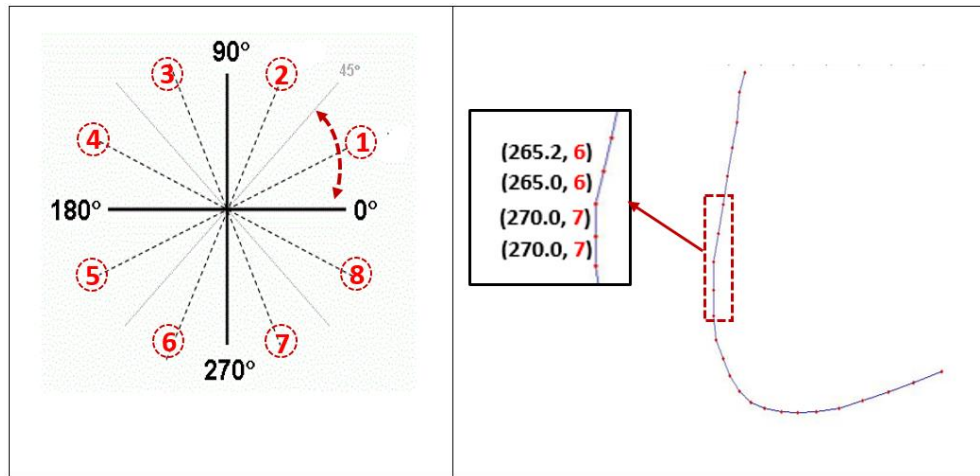


Figure 5-7 Angle Quantization feature

5.2.3.2 Orientation Histogram-based Feature

This feature is based on the histogram of grapheme's segments orientation. The details of this feature is presented in Chapter 3 (section. 3.3.3)

5.2.3.3 Curliness feature

This feature describes the curving of the grapheme (i.e. the grapheme points' deviation from a straight line). More details are presented in Chapter 4 (section 4.5.5).

5.2.3.4 Curvature and Writing Direction Feature

The curvature of a grapheme's trajectory at a specific point is a measure of how sensitive the tangent line of that point to moving it to other adjacent points [91]. The curvature at point $(x(t), y(t))$ is represented using the following sine and cosine functions:

$$\text{Cos}(\beta t) = \cos(\alpha(t-1)) * \cos \alpha(t+1) + \sin(\alpha(t-1)) * \sin(\alpha(t+1)),$$

$$\text{Sin}(\beta t) = \cos(\alpha(t-1)) * \sin(\alpha(t+1)) - \sin(\alpha(t-1)) * \cos(\alpha(t+1))$$

Where the values of $\cos \alpha$ and $\sin \alpha$ represent the writing direction and are computed as follows:

$$\sin(\alpha t) = \frac{\delta y(t)}{\delta s(t)}, \text{ and } \cos(\alpha t) = \frac{\delta x(t)}{\delta s(t)}, \text{ where}$$

$$\begin{aligned} \delta x(t) &= x(t-1) - x(t), \\ \delta y(t) &= y(t-1) - y(t), \text{ and} \\ \delta s(t) &= \sqrt{\delta x^2(t) + \delta y^2(t)} \end{aligned}$$

The difference between the curvatures and the writing direction is shown in figure 5-8 [97] [88].

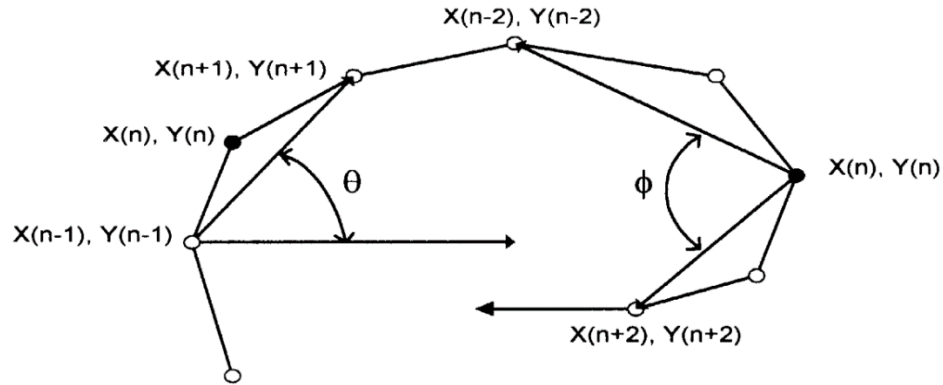


Figure 5-8 The difference between the curvature (\emptyset) and Writing direction (Θ)[88]

5.2.3.5 Polar Angular feature

This feature is based on the relationship between the point in the center of the bounding box surrounding the grapheme and all the grapheme points. More details and examples are presented in Chapter 4 (section 4.5.3).

5.2.3.6 End-Points Feature

This feature represents the grapheme based on the lengths and the directions of the lines connecting the first point, the middle point, and the last point of the grapheme. More specifically, this feature describes the following information: The lengths/directions of the lines connecting the start-point to middle-point, the end-point to middle-point, and the start-point to end-point of the grapheme.

Figure 5-9 shows two different graphemes with the lines connecting the start point, the middle point and the end point.

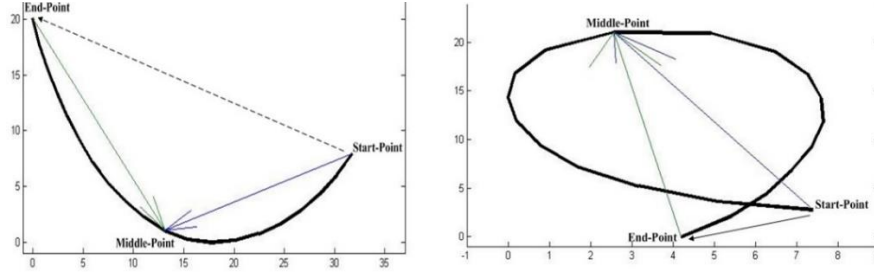


Figure 5-9 Grapheme's End-Points Feature

5.2.3.7 Loop detection feature

This feature indicates whether the grapheme points form a loop. The values of this feature are Boolean expressed as true or false.

5.2.3.8 Shape features

The following shape features are also used to classify the different graphemes' classes; the grapheme's length, width, height, and aspect ratio (represents the height to width ratio for the grapheme's trajectory).

5.2.4 Graphemes' Codebook Generation

The process of generating the graphemes' codebook from the extracted graphemes have been described in details in Chapter 4 (section 4.6). In brief, the process here is started by representing the training graphemes using the features described in the previous section (i.e. Angles quantization, orientation histogram-based, curliness, polar angular, curvature and writing direction, loop existence, length, width, height and aspect ratio). Then *k-means* clustering algorithm is used to cluster the graphemes into different clusters based on the used features. The representative samples of these clusters are used to build the graphemes' codebook. Figure 5-10 shows the graphemes' codebook generated in this process.

| Graphemes' Codebook | | | | | | | |
|---------------------|----|----|----|----|----|----|----|
| | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | | | | | | | |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| | | | | | | | |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| | | | | | | | |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| | | | | | | | |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| | | | | | | | |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| | | | | | | | |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| | | | | | | | |
| 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |

Figure 5-10 Graphemes' Codebook

The generated codebook is reduced by grouping similar classes into the same *bag-of-classes*. Therefore, each bag-of-classes contains a class or a set of classes that are similar.

Figure 5-11 shows the reduced codebook (*Bag-of-classes*).

| Graphemes' Codebook (Bag Of Classes) | | | | | | |
|--------------------------------------|----|----|----|----|----|----|
| | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | | | | | | |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| | | | | | | |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| | | | | | | |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| | | | | | | |
| 29 | 30 | 31 | 32 | 33 | 34 | 35 |

Figure 5-11 Reduced Codebook (Bag-of-Classes)

5.2.5 Feature Selection Approach

Selecting the most relevant and representative features is essential to achieve better accuracy. In our work, different features are designed and used in modelling the graphemes. The proposed approach for Arabic online text recognition was evaluated first using all features. Then, an approach based on ranking and merging the features is applied to select the most suitable features as follows. Initially, we create feature groups that have a single feature (i.e. each single feature forms a group). In the ranking phase, each group is evaluated alone for classification. Then the groups are ranked based on their achieved accuracies. The merge phase starts by considering the first ranked feature group as the selected feature group. Then the performance of using the second ranked feature group with the selected feature group is evaluated. If the accuracy is improved then the second ranked feature group is merged into the selected feature group. Otherwise, the second ranked feature group is ignored. This evaluating and merging process is repeated for all ranked feature groups till we get the final selected features.

As a result of this approach, the selected features are; curliness, curvature, writing direction, end-points, existence of loop, length, width, height and aspect ratio.

5.2.6 Codebook improvement based on the selected features

The graphemes' codebook is improved based on the selected features as follows. The extracted graphemes are represented using the selected features then *k-mean* clustering algorithm is used to cluster the graphemes based on these features. The initial codebook is generated from the representative samples of all generated clusters. The graphemes'

codebook is improved automatically according to inter-class similarities between the codebook elements (classes). Similar classes will be merge into the same *bag-of-classes* such that each bag contains a class or a set of classes that are similar. Finally, the grouping is refined subjectively to improve the generated Bag-of-classes. Figure 5-12 shows the generated codebook using the selected features. Arrows have been added in the figure to show the graphemes' directions.

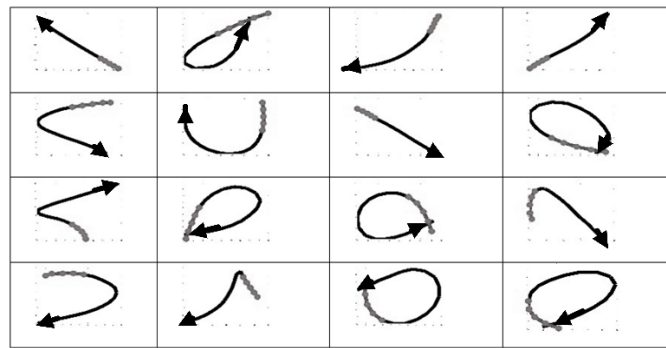


Figure 5-12 Codebook generated from the selected features

5.2.7 Fuzzy Modeling (generating graphemes' fuzzy models)

To address the problem of variability in writing, fuzzy models are generated for the extracted graphemes from the training characters. In order to generate the fuzzy models, the lengths of all graphemes are normalized to a unique length. In our work, we normalized the lengths to 20. This length is derived based on the common lengths of the graphemes. After normalizing the graphemes' lengths, the following features are used to represent the graphemes: curliness, curvature, writing direction, end-points, loop existence, length, width, height and aspect ratio (these features are described in section 5.2.3). These features are used to build generic models that represent the graphemes (i.e. fuzzy models) as

follows. For each extracted grapheme, the fuzzy model generated for that grapheme is a series of points that represent the features of that grapheme with the standard deviation (σ) of y-values of all the training graphemes from the same class at each point. The standard deviation for each class represents the fuzzy tolerance of the class's model and it is calculated from the graphemes samples of that class. The width of the fuzzy tolerance varies from one point to another according to the model as estimated from the training graphemes. In the recognition phase, computing the membership value for a testing sample at a certain sampling point is determined based on the width of the fuzzy tolerance. Therefore, we are using membership functions based on automatically generated duration and not fixed. This illustrates the robustness of the proposed fuzzy models in handling the variability of handwriting styles.

5.3 Recognition phase

In this phase, Arabic online text is modeled as a sequence of graphemes which represent the basic parts that text is composed of. Subsequently, these graphemes are recognized based on the fuzzy models generated in the training phase. The recognition process consist of different steps. The details of these steps are presented in the next subsections.

5.3.1 Preprocessing

After reading the Arabic text, the preprocessing steps are applied in order to enhance the text's trajectory and achieve better accuracy. In our work, removing duplication, simplification and smoothing are used in the preprocessing phase. These steps are

described in the training phase (section 5.2.1). Figure 5-13 shows the text “فصار طاهراً” before and after the preprocessing phase.

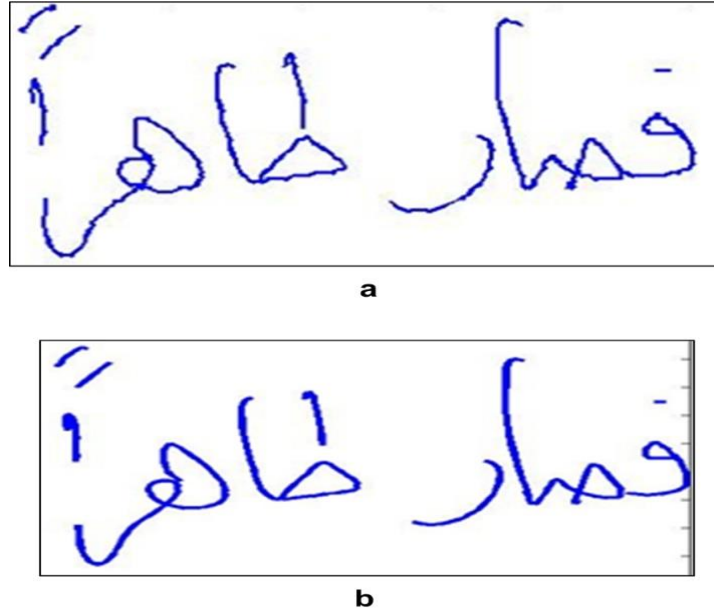


Figure 5-13 Arabic Online text line (a) before preprocessing (b) after preprocessing

5.3.2 Baseline Estimation

The baseline is the horizontal line on which the characters of a text line are joined. Estimating the baseline is very important for segmentation and feature extraction steps [98]. Different approaches have been proposed for estimating the baseline of the Arabic text [99]. Among these approach, the horizontal histogram projection is commonly used to estimate the Arabic text baseline [100]. In our work, we use horizontal histogram projection for estimating the baseline of Arabic online text. In addition to baseline, top-line, bottom-line, upper-median and lower-median lines are also estimated. Figure 5-14 shows the estimated base line for the text line “فصار طاهراً”.

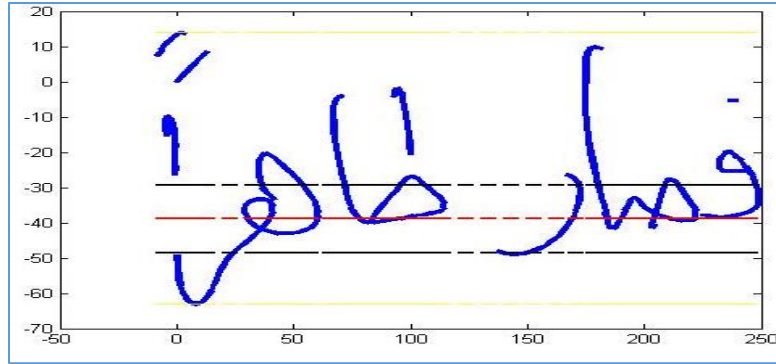


Figure 5-14 Baseline Estimation

5.3.3 Delayed strokes handling

In Arabic text, delayed strokes (sometimes called additional strokes or secondary's) are written above, below or within the characters. The appropriate handling of the delayed strokes is essential for adequate recognition rate. Although the size of the delayed strokes, especially the dots, is usually small compared with other strokes, this feature alone is not enough for detecting all types of delayed strokes. For example, figure 5-15 shows an Arabic word where the delayed stroke of the letter TAA has greater size than the main stroke of the letter ALEF.

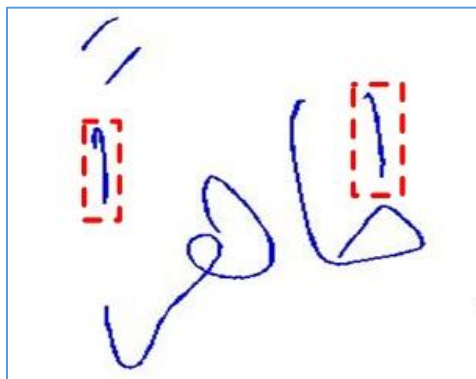


Figure 5-15 The delayed stroke of letter TAA has grater size than the main stroke of letter ALEF

Furthermore, estimating the text's height is important to detect the delayed strokes. In our work, and in order to estimate the line's height, a window is drawn around each stroke, as shown in figure 5-16, then the heights of all windows are calculated. The maximum height, *MaxHeight*, will be used as the estimated line's height.

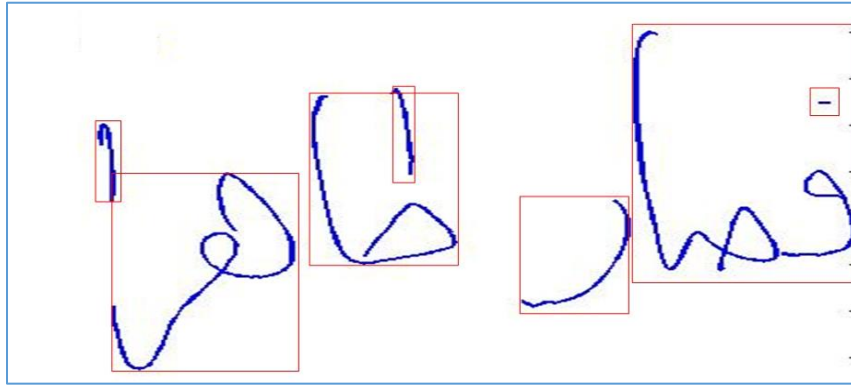


Figure 5-16 Text line with windows drawn around its strokes

In addition to the stroke's size and height, the following features are also used to detect the delayed strokes: the height/width ratio and the overlap between strokes. These features need to be combined in order to get better results. For example, in some cases and due to the variation in handwriting, we could have a delayed stroke that has a height greater than the height of a main stroke as shown in figure 5-17.a. Combining the height with the overlap feature resolves such cases.

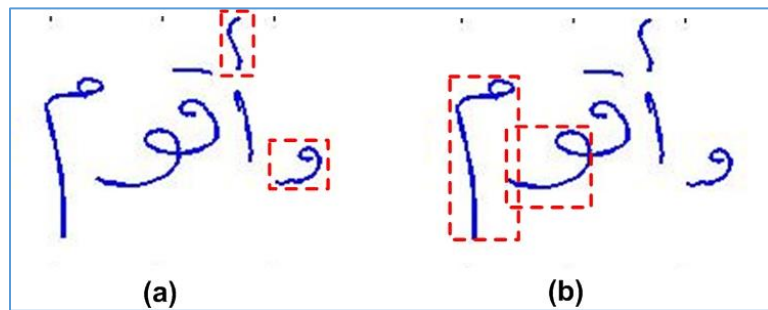


Figure 5-17 (a) The height of the delayed stroke of letter ALEF is greater than the height of letter WAW. (b) Partial overlap between the main strokes of letters WAW and MEEM.

The overlap between strokes will be used to detect the delayed strokes that are overlapped with their main strokes (usually with their previous strokes). In Arabic text, some letters may overlap partially with other letters as shown in figure 5-17.b. To avoid considering this partial overlap in the delayed stroke detection, a threshold, *ThresholdOverlap*, is used such that a stroke is detected as a delayed stroke if a complete overlap occurs or the amount of overlap $> ThresholdOverlap$.

The algorithm *DelayStrokesDetection* presents the main steps used for detecting the delayed strokes.

Algorithm *DelayStrokesDetection*

1. Find the Line's height (*MaxHeight* of all strokes).
2. $MainStrokes \leftarrow \{\}$, $DelayedStrokes \leftarrow \{\}$
3. **Repeat for all strokes**
 - If (stroke's size $< Threshold_{Size}$) and (Height/width ratio $< Threshold_{Ratio}$)
and (Height $< Threshold_{Height} * \text{Line's height}$)
 $DelayedStrokes \leftarrow \text{stroke}$
 - Else
 - If (overlap exists) and (overlap $> Threshold_{Overlap}$)
 $DelayedStrokes \leftarrow \text{stroke}$
 - Else
 $MainStrokes \leftarrow \text{stroke}$
4. **Until the last stroke in the text is reached**

5.3.4 Graphemes' Extraction

In this section we describe the proposed algorithm for extracting the graphemes from the Arabic online text. The algorithm *TextGraphemeExtraction* below presents the different steps used for extracting the line's graphemes.

Algorithm *TextGraphemeExtraction*

1. Find the Loops.
2. Find the Open-loops.
3. Find Possible Segmentation Points (PSPs).
 - *Using Curvature of the text's trajectory.*
4. Filter PSPs:
 - Rule 1: Excluding PSPs that are inside loops.
 - Rule 2: Excluding PSPs that are far away from the baseline
 - *Exclude if $PSP > (Upper-Median + Threshold)$.*
 - *Exclude if $PSP < (Lower-Median - Threshold)$.*

The steps of the proposed algorithm will be described in detail in the following subsections.

5.3.4.1 Loop detection

There are nine Arabic letters that contain loops (ه, و, م, ق, ف, ظ, ط, ض, ص) and five letters that some writers write with loops (غ, ع, ج, ح, خ). This makes the existence of loop a useful feature for recognizing Arabic letters. Moreover, the loop can be considered as a basic grapheme in modeling its letter. Figure 5-18 shows some Arabic online characters with loops.

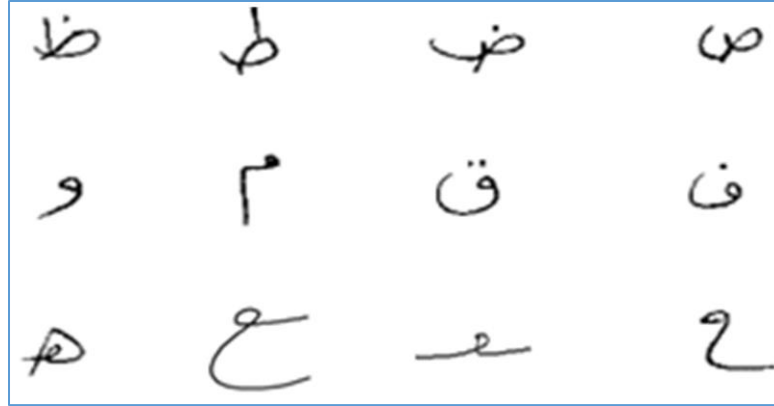


Figure 5-18 Different Arabic online characters with loops

In our approach, the loop is detected and its points are extracted to be used as a basic grapheme for the letter. Closed loop detection is done by finding the self-intersection in the trajectory. Figure 5-19 shows the line (فصار طاهرا) after detecting the loops of letters FAA, SAD and HAA. The loop of letter TAA is not detected at this level since it is an open loop as shown in the zoomed part in the figure.

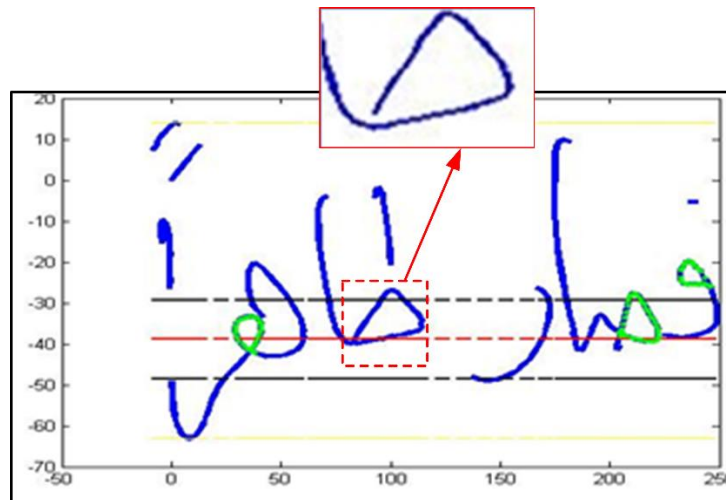


Figure 5-19 The text line after finding the loops

5.3.4.2 Open loop detection

The loops are sometimes written as open-loops as shown in figure 5-20.a. This creates a problem in detecting these loops and recognizing the corresponding letters. To solve this problem we close these open-loops by applying the following steps:

1. Finding the direction of the first n points in the stroke.
2. Adding n points at the beginning of the stroke (before the first point) according to the detected direction.
3. Checking if there is self-intersection after adding these points.

Figure 5-20 shows the steps of closing the loop of letter WAW.

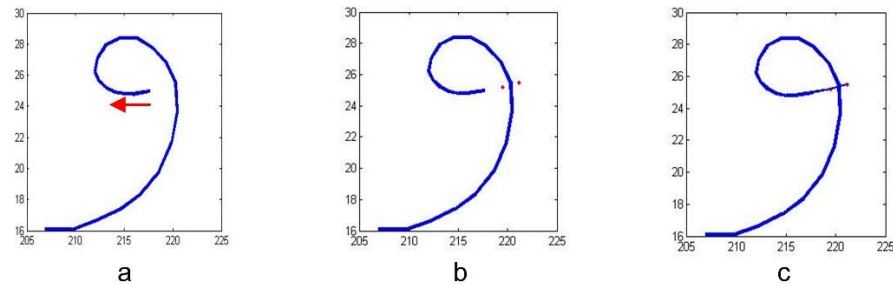


Figure 5-20 Closing Open-loop

Figure 5-21 shows the line (فصار طاهرا) after closing the open-loop of letter TAA.

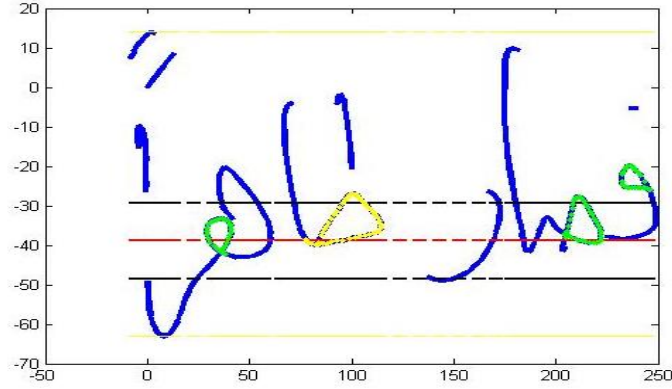


Figure 5-21 Text line after closing the loop of letter TAA

5.3.4.3 Finding Possible Segmentation Points (PSP)

The Possible Segmentation points (PSPs) are detected by measuring the curvature of the text's trajectory; if the curvature is greater than a *threshold* then that is a candidate PSP.

The detected PSPs in the line (فصار طاهرا) are shown in figure 5-22.

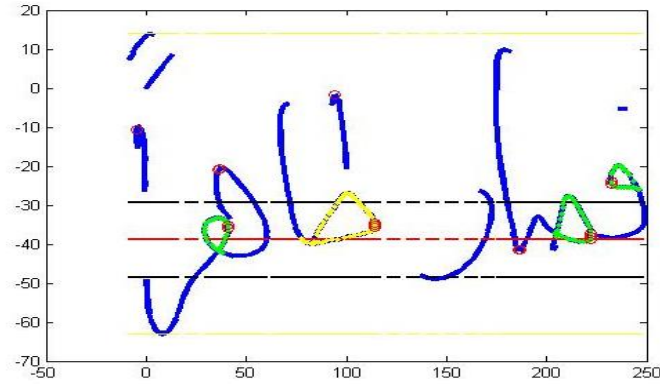


Figure 5-22 Possible segmentation points

5.3.4.4 Filtering Possible Segmentation Points (PSPs)

The detected PSPs are filtered using the following rules:

Rule 1: *Excluding PSPs that are at the loops.*

If the selected PSP is inside a loop then that point will be excluded from the PSPs list since the whole loop points represent one grapheme and is not segmented into more than one grapheme.

Rule 2: Excluding PSPs that are far away from the baseline

Based on the nature of Arabic text, characters are connected at the baseline. Most of the proposed segmentation approaches in the literature assume that the Arabic characters are connected at the baseline [8]. Graphemes' cut points are also located close to the baseline. Therefore the PSP that is far from the baseline most likely will not be a cut point. In our approach, the distance between the candidate PSP and the baseline is measured and if that distance is greater than a threshold then that PSP will be excluded from the PSP list. Specifically, the PSP is excluded if:

$$\text{PSP} > (\text{Upper-Median} + \text{Threshold}), \text{ or}$$

$$\text{PSP} < (\text{Lower-Median} - \text{Threshold}).$$

5.3.5 Feature Extraction

The length of the extracted graphemes are normalized by adjusting all the graphemes' lengths to 20 points. Then the normalized graphemes are represented using the features described in the training phase, section 5.2.3, (viz. curliness, curvature, writing direction, end-points, loop existence, length, width, height and aspect ratio).

5.3.6 The graphemes' relative height feature (Grapheme/Line height ratio)

In Arabic text, the relative height of the characters is important and can be used to resolve the confusion between the different characters that may look similar. As a result, the relative height of the graphemes, extracted from the characters, represents useful information which may help in differentiating between similar graphemes. The normalization process may destroy this useful information and may thus cause confusion between samples that look similar. In our work, the estimation of the line's height is done by drawing a window around each stroke, as shown in figure 5-23.b, then the heights of all windows are calculated. The maximum height are used as the estimated line's height. This approach is more accurate than the estimation of the height using one window as shown in figure 5-23.a. The relative height of the grapheme is defined as grapheme's height/ line's height. Table 5-2 shows the relative heights of the Arabic characters estimated from the training characters.

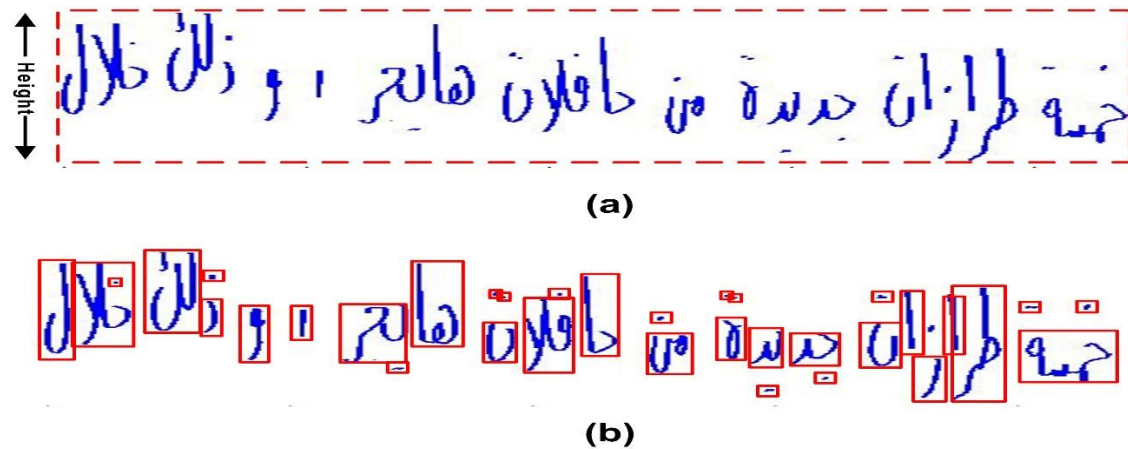


Figure 5-23 Estimating the height of the Text Line (a) Height's Estimation based on using one window (b) Estimation based on using a window for each stroke and taking the height of the maximum one

Table 5-2 Relative Heights of Arabic characters

| Character | | Height Ratio | Character | | Height Ratio | Character | | Height Ratio |
|-----------|----|--------------|-----------|----|--------------|-----------|----|--------------|
| B_al | أ | 0.28739 | E_ja | ج | 0.652174 | I_laaa | لا | 0.731522 |
| B_ay | ع | 0.542857 | E_ka | ق | 0.483485 | I_laae | لا | 0.666356 |
| B_ba | ب | 0.320313 | E_ke | ك | 0.720674 | I_laah | لا | 0.802335 |
| B_de | ض | 0.442648 | E_kh | خ | 0.67439 | I_lae | لي | 0.704232 |
| B_fa | ف | 0.325659 | E_la | ل | 0.835244 | I_ma | م | 0.75742 |
| B_gh | غ | 0.241379 | E_laaa | لا | 0.683855 | I_na | ن | 0.404509 |
| B_ha | ح | 0.265432 | E_ma | م | 0.70592 | I_ra | ر | 0.351247 |
| B_he | ه | 0.337073 | E_na | ن | 0.454695 | I_se | س | 0.507173 |
| B_ja | ج | 0.225883 | E_ra | ر | 0.380893 | I_ta | ت | 0.414232 |
| B_ka | ك | 0.28081 | E_sa | ص | 0.525238 | I_tee | ة | 0.307848 |
| B_ke | ك | 0.6324 | E_se | س | 0.459514 | I_th | ث | 0.35269 |
| B_kh | خ | 0.237853 | E_sh | ش | 0.513732 | I_wa | و | 0.349286 |
| B_la | ل | 0.540667 | E_ta | ت | 0.37628 | I_wl | ؤ | 0.413492 |
| B_laha | لح | 0.692737 | E_tee | ة | 0.38423 | I_ya | ي | 0.540847 |
| B_lama | لم | 0.56358 | E_th | ث | 0.416198 | I_za | ز | 0.337287 |
| B_ma | م | 0.198569 | E_to | ط | 0.370863 | M_al | أ | 0.120537 |
| B_na | ن | 0.323663 | E_wa | و | 0.378069 | M_ay | ع | 0.257408 |
| B_sa | ص | 0.435779 | E_wl | ؤ | 0.415284 | M_ba | ب | 0.150194 |
| B_se | س | 0.235686 | E_ya | ي | 0.488303 | M_de | ض | 0.348417 |
| B_sh | ش | 0.259126 | E_za | ز | 0.455282 | M_fa | ف | 0.233173 |
| B_ta | ت | 0.273617 | I_aa | ا | 0.422224 | M_gh | غ | 0.267029 |
| B_th | ث | 0.192938 | I_ae | أ | 0.444123 | M_ha | ح | 0.254406 |
| B_to | ط | 0.480462 | I_ah | إ | 0.474016 | M_he | ه | 0.383133 |
| B_ya | ي | 0.300138 | I_am | أ | 0.494867 | M_ja | ج | 0.211695 |
| B_zha | ظ | 0.462652 | I_ay | ع | 0.800824 | M_ka | ك | 0.237956 |
| E_aa | ا | 0.552771 | I_ba | ب | 0.293662 | M_ke | ك | 0.685975 |
| E_ae | أ | 0.514772 | I_da | د | 0.286077 | M_kh | خ | 0.212475 |
| E_ah | إ | 0.536445 | I_de | ض | 0.528122 | M_la | ل | 0.555395 |
| E_ay | ع | 0.827303 | I_dh | ذ | 0.28736 | M_ma | م | 0.203839 |
| E_ba | ب | 0.412795 | I_ee | ي | 0.444493 | M_na | ن | 0.152068 |
| E_da | د | 0.266933 | I_fa | ف | 0.391612 | M_sa | ص | 0.378894 |
| E_de | ض | 0.461424 | I_ha | ح | 0.854018 | M_se | س | 0.157478 |
| E_dh | ذ | 0.248167 | I_he | ه | 0.283071 | M_sh | ش | 0.156308 |
| E_ee | ي | 0.43014 | I_hh | ه | 0.320499 | M_ta | ت | 0.148432 |
| E_fa | ف | 0.423775 | I_ja | ج | 0.832702 | M_th | ث | 0.175774 |
| E_gh | غ | 0.731487 | I_ka | ق | 0.516145 | M_to | ط | 0.360625 |
| E_ha | ح | 0.632163 | I_ke | ك | 0.688681 | M_ya | ع | 0.181399 |
| E_he | ه | 0.376598 | I_la | ل | 0.779594 | M_zha | ظ | 0.299152 |

5.4 Arabic Online text line recognition

Figure 5-24 shows the proposed steps for recognizing Arabic online text. Initially, the online text is segmented into its basic graphemes using the algorithm *TextGraphemeExtraction* described in section 5.3.4. Then the extracted graphemes are classified using *FuzzyClassification* algorithm to find the most similar class for each grapheme. After that, the recognized graphemes are mapped to the corresponding characters using *MapGraphemesToCharacter* algorithm. These steps are addressed in details in following sections.

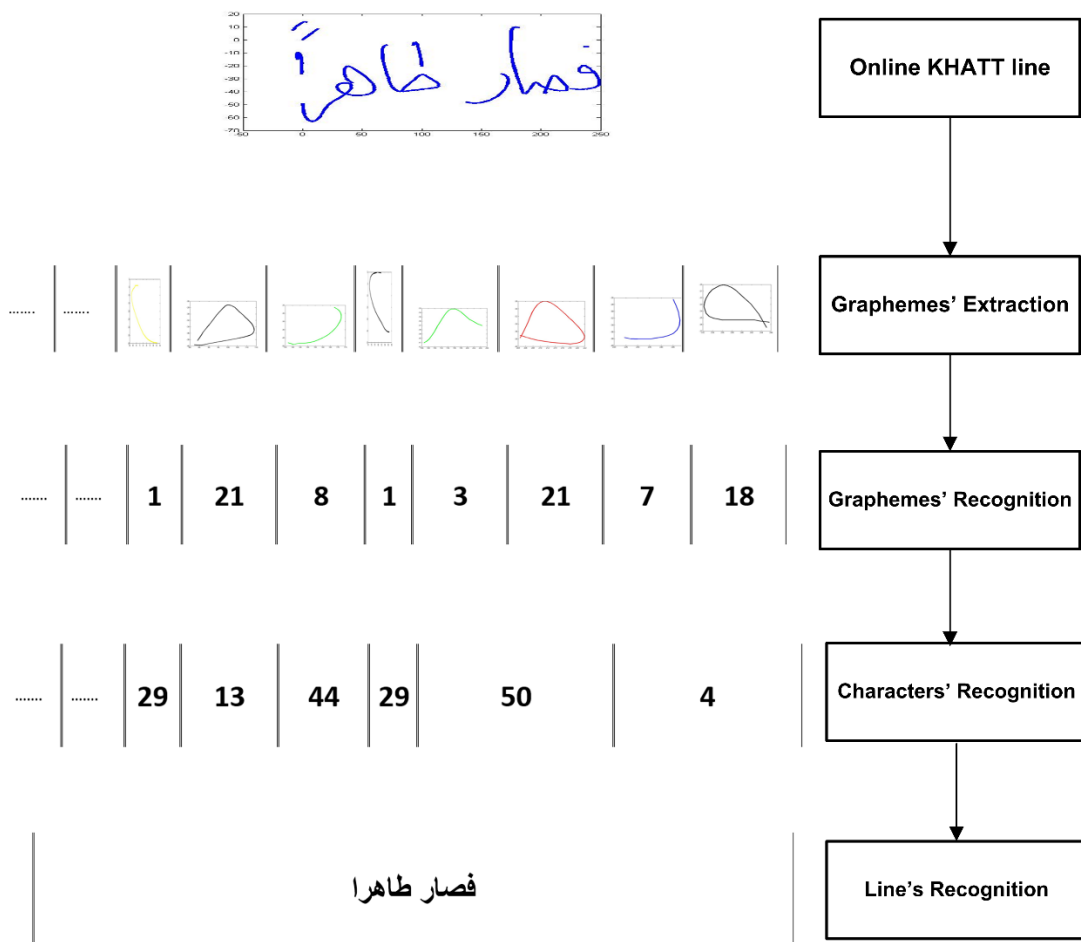


Figure 5-24 Illustration of steps to recognize Arabic Online line from Online KHATT database

5.4.1 Graphemes' Recognition (Fuzzy Classification)

After extracting the text's graphemes, we use fuzzy classification to recognize the extracted graphemes using the fuzzy models generated from the graphemes of the training characters. For each extracted grapheme, fuzzy comparisons between the grapheme and the models are done to find the most similar one as follows. At each point in the grapheme, fuzzy comparison is performed to estimate the membership between the grapheme and the model. The membership value between a grapheme's point g and the model at sampling point m is estimated base on the following equation:

$$Mv(g, m) = \begin{cases} 0, & g > \alpha_1 \text{ or } g < \alpha_2 & (a) \\ 1, & \beta_2 \leq g \leq \beta_1 & (b) \\ \frac{\alpha_1 - g}{\alpha_1 - \beta_1}, & \beta_1 < g \leq \alpha_1 & (c) \\ \frac{g - \alpha_2}{\beta_2 - \alpha_2}, & \alpha_2 \leq g < \beta_2 & (d) \end{cases}$$

Where $(\beta_1$ and $\beta_2)$ represent the first tolerance region and $(\alpha_1$ and $\alpha_2)$ represent the second tolerance region as shown in the trapezoidal membership function in figure 5-25. The values of β_1 , β_2 , α_1 and α_2 are automatically generated from the training data.

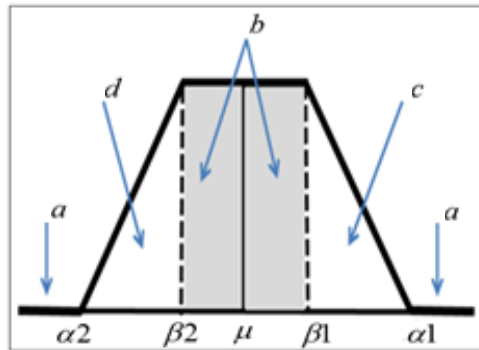


Figure 5-25 Trapezoidal membership function

The membership value (M_v) represents the weight of the similarity between the grapheme and the model at sampling point m . The value of $M_v \in [0, 1]$ such that ‘0’ (part (a) in the Equation) means least similarity between the grapheme and the model at that sampling point. Membership value ‘1’ (part (b) in the Equation) represents the maximum similarity. Membership value in the regions between (α_1 and β_1) and between (β_2 and α_2) are calculated using parts (c) and (d) of the Equation respectively. The overall similarity between a grapheme G and a model M is given by

$$\text{Similarity}(G, M) = \frac{1}{N} \sum_{i=1}^N M_v (G(i), M(i)),$$

Where N is the number of points. The model with the highest similarity will be selected as the recognized grapheme’s class.

5.4.2 Character Recognition (Graph-based Classification)

The previous phase, graphemes’ recognition, results in a list of model numbers that represent the recognized graphemes’ classes. Each number, or sequence of numbers, is corresponding to a specific character and has to be mapped to that character during the character recognition phase. In order to do this mapping, statistics about all Arabic characters and their corresponding grapheme classes’ numbers are needed. These statistics are gathered by analyzing the training characters. For each character, the probability of representing the character by a grapheme (or a sequence of graphemes) is calculated and used in mapping the grapheme (or a sequence of graphemes) to its (their) corresponding character.

5.4.2.1 Character Grapheme-based Probability

Due to variations in handwriting, different occurrences of the same character may be represented by different sequences of graphemes. Moreover, different characters may have common sub sequence of graphemes. In order to map each grapheme (or sequence of graphemes) to its (their) corresponding character, we compute the grapheme-based probability for each character as follows.

Let $GS = \{gs_1, gs_2, \dots, gs_n\}$ be a set of all grapheme sequences extracted from the training data. The conditional probability of a character $Char_i$ represented by a sequence $gs_i \in GS$ is defined as follow:

$$P(Char_i | gs_j) = \frac{C(Char_i, gs_j)}{C(gs_j)}$$

Where $C(Char_i, gs_j)$ is the number of occurrences of the character $Char_i$ represented using the grapheme sequence gs_i and $C(gs_j)$ is the number of occurrences of all characters represented using the grapheme sequence gs_i .

5.4.2.2 Graph Construction

Our proposed character recognition approach is based on construction of a graph where nodes represent Grapheme Patterns (candidate characters) and edges are created between nodes depending on the grapheme-based probabilities calculated from the training data. The *Grapheme Pattern* is defined as follows.

Grapheme Pattern (GP): Given the input text T , let $GraphemeList = \{G_1, G_2, G_3 \dots G_{n-1}, G_n\}$, be a sequence of graphemes extracted from T where n is the number of graphemes. For each grapheme $G_i \in GraphemeList$, the Grapheme Patterns (GP) of G_i , $GP(G_i)$, is a set of all candidate sequences of graphemes that start with G_i and their length is less than or equal to m where m is the maximum possible number of graphemes in a character in the training set. For example:

- $GP(G_1) = \{G_1, G_1G_2, G_1G_2G_3\dots\}$, i.e. patterns for characters that start with grapheme G_1 .
- $GP(G_2) = \{G_2, G_2G_3, G_2G_3G_4\dots\}$, i.e. patterns for characters that start with grapheme G_2 .
- ...
- ...
- and so on.

For a given $GraphemeList G = \{G_1, G_2, G_3 \dots G_{n-1}, G_n\}$, extracted from a text T , the process of constructing the graph starts by generating all Grapheme Patterns (GPs) of the first grapheme, G_1 . For each generated grapheme pattern (GP), which is corresponding to a specific character C_i , we find its probability P_i based on the grapheme-based probabilities calculated from the training data. Then all generated grapheme patterns are added as nodes in the first level of the graph and connected to the root node and their probabilities are assigned to the connected edges. After that, for each added node n , we find the last grapheme, *last*; then the subsequent grapheme of *last* is used to generate a new set of grapheme patterns. These generated grapheme patterns are added as nodes in the next level

of the graph and connected to the node n and their probabilities are assigned to the connected edges. This process is repeated until the last grapheme, *last*, in each node (in the last level of the graph) has no subsequent grapheme in the *GraphemeList* i.e. *last* is equal to G_n .

The algorithm *MapGraphemesToCharacter* shows the main steps of our graph-based approach proposed to find the corresponding character for a grapheme (or a sequence of graphemes). Note that, the algorithm generates the nodes located in the first level of the graph in step 3 and connects them to the root node, whereas the loop in steps 4-5 is responsible for generating the nodes in the subsequent levels of the graph.

5.4.2.3 Path evaluation

Nodes in the graph represent grapheme patterns which correspond to candidate recognized characters, while edges are created between nodes depending on the characters grapheme-probabilities. Each path in the graph represents a possible solution for the input text. Among these paths, we need to find the best one that represents the recognized text. Therefore, all paths are evaluated using the following formula:

$$C = \sum_{k=2}^n \ln(P_k))$$

Where C is the path probability, n is the number of nodes in the path and P_k is the probability assigned to the edge $e : (n_{k-1}, n_k)$, i.e. the edge between nodes n_{k-1} and n_k . The characters corresponding to the nodes in the path with the maximum probability will be selected as the recognized text.

Algorithm *MapGraphemesToCharacter*

1. Let *GraphemeList* = $\{G_1, G_2, G_3 \dots G_{n-1}, G_n\}$, be a sequence of graphemes extracted from the text *T*.
2. Generate all grapheme patterns (*GPs*) of the first grapheme G_1 . (*GPs represent the candidate characters*)
3. For each generated Grapheme Pattern, $gp_i \in GPs$,
 - a. Find its probability P_i based on the grapheme-based probabilities calculated from the training data.
 - b. Create a node n and add it to the graph.
 - c. Connect n to the root node.
 - d. Assign P_i as a weight to the connected edges.
4. **Repeat**
For each node n in the current level
 - a. Find the last grapheme, *last*, in the node n
 - b. Find the subsequent grapheme, *Subseq*, of *last*.
 - c. Generate a set of grapheme patterns *S* from *Subseq*.
 - d. Create a node for each grapheme pattern in *S* and add it to the graph.
 - e. Connect each generated node to the node n and assign its probability P_i as a weight to the connected edge.
5. **Until** last grapheme, *last*, in each node, in the last level of the graph, has no subsequent grapheme in the grapheme list *G*.

Figure 5-26 shows an example of the initial steps for mapping the recognized graphemes to their corresponding characters for the given text. The characters connected to the grapheme pattern in the figure represent the candidate characters corresponding to that grapheme pattern and the number attached with the character represents the character's probability. The sign \emptyset means that the *Grapheme Pattern* does not represent any character.

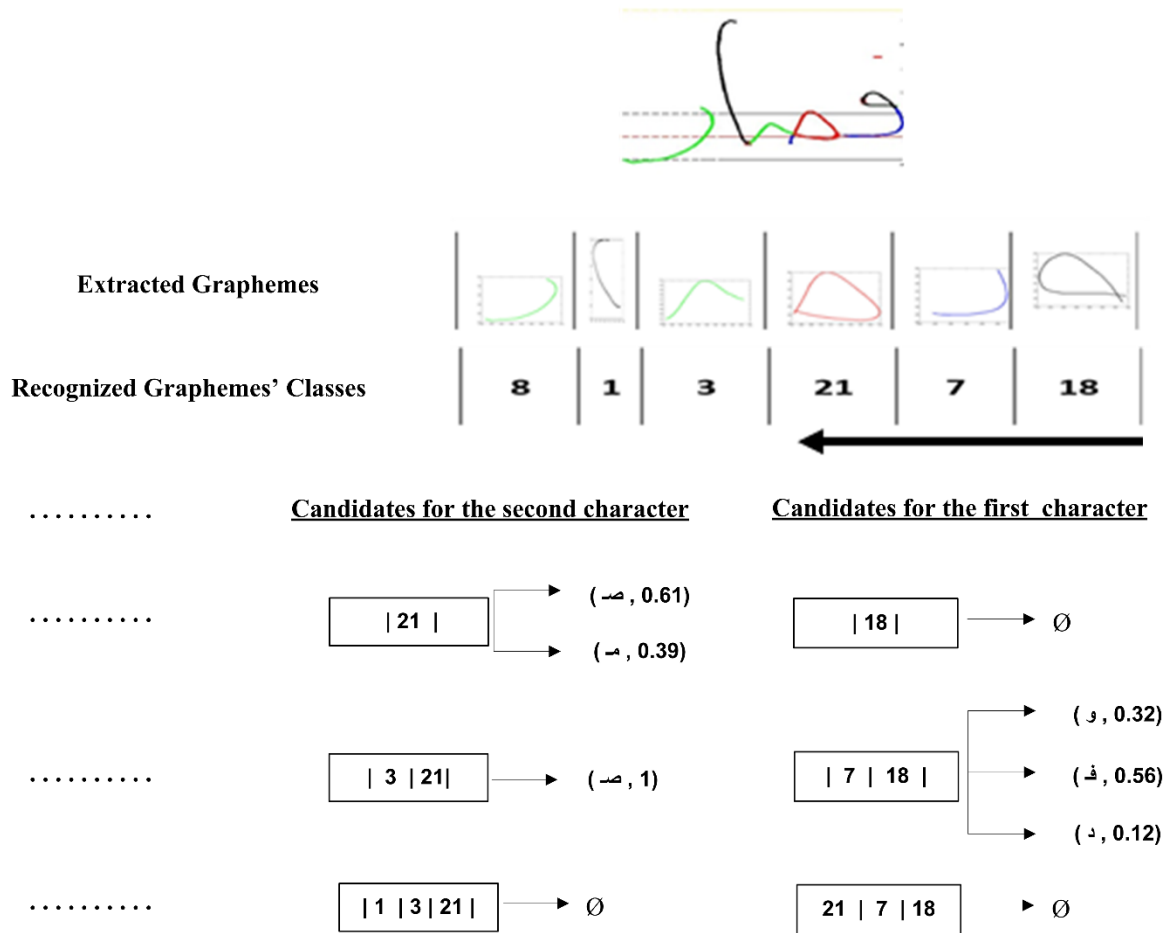


Figure 5-26 Example to show the steps of mapping the graphemes to their corresponding characters

Figure 5-27 shows the graph constructed for the previous example. The nodes in the graph represent the candidate characters while numbers between nodes represent the probabilities of these characters. These probabilities are calculated based on statistics gathered from the training characters. Each path in the graph represents a possible solution for the text. Our approach selects the one with the maximum probability as the recognized text.

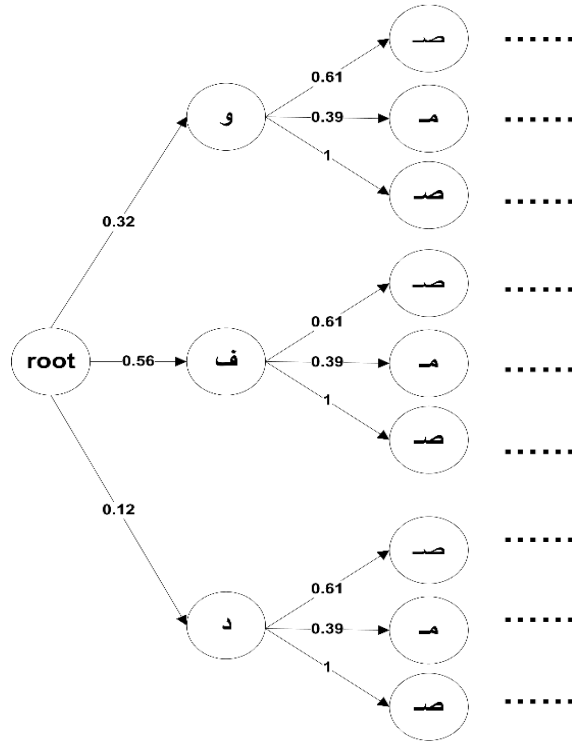


Figure 5-27 Initial steps of constructing a graph for recognizing a text line

5.5 Experimental Work

In this section we present our experimental work with Arabic online text. The details of the experiments and the used databases are discussed.

5.5.1 Grapheme's fuzzy models Generation

As we have discussed before in section 5.2, the graphemes' fuzzy models in the training phase are built from isolated characters. To build models that better address the problems of characters' connections in the Arabic online text, we decide to use pre-segmented Arabic online characters. For this purpose, a data set containing 3882 pre-segmented Arabic online

characters is used. These characters are manually segmented from *Online-KHATT* database and include samples from all Arabic characters in their different shapes.

In our work, the relative height ratios are utilized by integrating them into the graphemes' fuzzy models. The relative height ratios are calculated for the training graphemes according to the heights of the original lines from which the training characters are segmented. In the testing phase, the relative height ratio features are calculated and assigned more weight than other features during the similarity calculation.

5.5.2 Performance evaluation

To evaluate the performance of our proposed approach, we use *HResult* tool provided by HTK [101]. It compares the output transcription file (produced by our classifier) with the corresponding original reference transcription file. *HResult* uses two measurements to evaluate the system performance which are the correctness and the accuracy. These measurements evaluate the percentage number of labels correctly recognized as follows:

$$\% \text{ Correctness} = \frac{N - D - S}{N} * 100$$

$$\% \text{ Accuracy} = \frac{N - D - S - I}{N} * 100$$

Where N is the total number of labels in the transcription files, D is the number of deletions, S is the number of substitutions and I is the number of insertions.

5.5.3 Experimentation using Arabic Online words

In this section, we present our experimental results on Arabic online words. For this purpose, we collected two hundred Arabic Online words from 10 different writers. The collected words are selected from the fixed paragraph that consists of minimal Arabic text covering the different shapes of Arabic alphabet in their all forms [102]. Figure 5-28 shows some samples from the collected words.

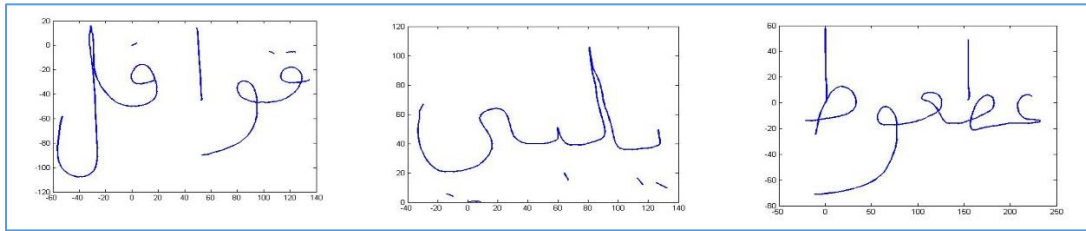


Figure 5-28 Some samples from the collected Arabic Online words.

Table 5-3 shows the obtained results after selecting the best features and reducing the codebook accordingly as described in sections 5.2.5 and 5.2.6. The table shows the results with different fuzzy tolerance values.

Table 5-3 Results of Online Words Recognition Using HResult tool

| Fuzzy Tolerance | %Correctness | %Accuracy |
|-----------------|--------------|--------------|
| 0.35 | 58.28 | 50.88 |
| 0.30 | 59.11 | 51.94 |
| 0.28 | 58.40 | 50.88 |
| 0.25 | 58.05 | 50.41 |

As shown in the table, the best obtained correctness value is around 59%. These results are in terms of character recognition, which can be relatively considered as competitive results

compared with similar works on Arabic text using structural approaches and without using post-processing steps to improve the results.

5.5.4 Experimentation using *Online-KHATT* database

In this section, we present our experimental results for Arabic online text lines recognition. For this purpose, several lines were selected from *Online-KHATT* database which is a large Arabic online text database collected in KFUPM from more than 600 writers.

The selected text lines are divided into two disjoint sets. The validation set, *50 lines*, is used for fuzzy model parameters' estimation while the test set, *100 lines*, is used to test the performance of the proposed approach. We evaluated the performance of the proposed approach using the selected structural features as described in section 5.2.5. We also conducted the experiments using turning function. Table 5-4 reports the results of these experiments.

Table 5-4 Results of Online Text Lines Recognition Using HResult tool

| Representation | Top n | %Correctness | %Accuracy |
|-------------------|------------------------------|--------------|--------------|
| Selected Features | Top 1 (1 st Path) | 42.30 | 31.52 |
| | Top 5 (5 Paths) | 44.12 | 33.19 |
| Turning Function | Top 1 (1 st Path) | 35.35 | 24.28 |
| | Top 5 (5 Paths) | 37.34 | 26.68 |

These experiments were conducted without using a lexicon. However, it is expected that the obtained results will be improved by consulting a lexicon of Arabic words. Using a lexicon and a language model is part of the post-processing phase and can be regarded as a future work.

In addition, the fuzzy models for the training graphemes are built from a dataset of pre-segmented characters which is different from the text database we used for testing.

Therefore, the writers in the training database are different from writers of the testing data.

The obtained recognition rate depends on many factors. The following paragraphs discuss some issues that affect the accuracy of our Arabic online text line recognition system.

- The used data in these experiments is natural online handwriting text. Moreover, the writing itself is a subjective process and it has different styles. The writing quality is affected by writers' skills. In one experiment, when we choose the more readable lines from the selected 100 lines, the obtained correction reaches 50% as shown in table 5-5.

Table 5-5 Results of the more readable lines Using HResult tool

| Top n | %Correctness | %Accuracy |
|------------------------------|--------------|-----------|
| Top 1 (1 st Path) | 50.29 | 38.83 |
| Top 5 (5 Paths) | 51.46 | 41.36 |

- The text lines are selected from *Online-KHATT* database which is an open-vocabulary database (i.e. it has unlimited vocabularies). Most of the reported work is based on closed-vocabulary databases. In some work, a database with very limited words is used. For example, only one word "عزالدين" is replicated by the same writer in [103] and different samples from only two names (i.e. 'سلمى' and 'سحر') is used in [57].
- The loop represents a basic grapheme in our grapheme-based modelling approach. This makes the existence of loop a useful information for

recognizing the Arabic characters in the text. One of the peculiar trait of some writers is that, they draw loop for characters that don't require loop as shown in figure 5-29. Such cases can be a source of error and result in problems in recognizing the corresponding characters.

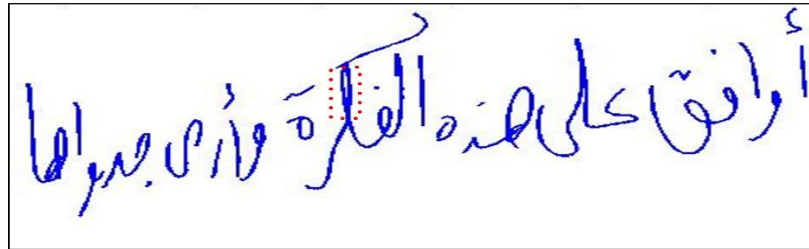


Figure 5-29 Loop is drawn with character that do not require loop

5.6 Conclusions

In this chapter, we have described the Arabic online text recognition approach proposed in this thesis. The proposed approach is based on segmenting the text into graphemes that can represent a whole character or a part from a characters' trajectory.

Pre-segmented Arabic online characters are used in the training phase to build the fuzzy models while Arabic online texts are used in the testing phase. Hence, in our approach the training and testing phases are independent which makes the proposed approach general and more suitable for handling unknown data. We choose to use a database of pre-segmented characters in the training phase instead of using isolated characters to better handle the issues related to characters' connections in an input text.

The training phase has two stages: codebook generation and fuzzy modeling. The codebook is generated from the extracted graphemes of the training characters. To better address the problem of variability in writing, the grapheme models are built in a fuzzy concept. The relative heights of the extracted graphemes are utilized by integrating them into the graphemes' fuzzy models.

Different features are used in modelling the extracted graphemes and an approach based on ranking and merging the features is applied to select the most suitable features.

In the recognition phase, the Arabic online text is modeled as a sequence of graphemes which represent its basic parts. For this purpose, a new approach for segmenting Arabic online text into its graphemes is proposed. The proposed approach starts by extracting the loops from the text since loops are considered as basic graphemes in modeling the Arabic text. For the non-loop trajectory's points, an initial set of Possible Segmentation Points (PSPs) is detected by measuring the curvature of the text's trajectory and finding the sharp turns in the trajectory. Then, a rule-based filtering algorithm is used to produce the final segmentation points (FSPs). The filtering algorithm utilizes the way characters are joined in Arabic online text.

A fuzzy classification approach is used to recognize the graphemes using the fuzzy models generated from the graphemes of the training characters. Then, the recognized graphemes are mapped to their corresponding characters based on graphemes' statistics gathered by analyzing the training characters.

CHAPTER 6

CONCLUSIONS AND FUTUER RESEARCH DIRECTIONS

In this chapter, we summarize the contributions of this thesis and discusses some issues relevant to the developed techniques. In addition, future research directions related to Arabic online text recognition are presented.

6.1 Conclusions

In recent years many researches have been conducted on the recognition of offline and online Latin text. Most of the research on Arabic addressed offline Arabic handwritten text while few addressed online Arabic text. The reasons for this may be due to the challenges related to online Arabic handwriting recognition systems. These challenges include the need for special hardware; the lack of freely available comprehensive databases for Arabic online text. In addition, the writing using electronic pen is less controlled than writing using a pen on paper which leads to more variability between writers, and even with the same writer.

Statistical-based approaches have been widely used in Arabic online text recognition research while structural-based approaches have remained comparatively unexplored. In this thesis, research on automatic recognition of online handwritten Arabic text using structural-based techniques has been conducted. We conducted research on issues relevant to automatic recognition of Arabic online digits, isolated characters, words and text.

A literature review of Arabic online text recognition is presented in chapter 2 that surveys the published work related to this thesis. It also includes a study of the different attempts to build a comprehensive database for Arabic online text.

An approach based on fuzzy modeling for the automatic recognition of Arabic online digits is presented in chapter 3. In this approach, fuzzy models of the different digits are automatically generated using the training data. The fuzzy intervals are generated automatically based on the analysis of the training samples at the digit segment level and not set manually at the digit level as was done in the previous works. In addition, we automatically generate weights for the different segments using the training samples. These weights are integrated in the fuzzy similarity estimate.

A two stage classification approach is implemented where SVM is used in the first stage and fuzzy-based classification using the automatically generated models in the second stage. The second stage has an integrated feedback verification step which verifies the recognized test sample label and possibly selects a better alternative. The proposed approach is evaluated using a database containing more than 30,000 Arabic online handwritten digit. An overall accuracy of 99.55% was achieved in the first stage (classify zero-nonzero digits) and the second stage classifies digits 1 to 9 with an accuracy of 98 %. This result, based on using our fuzzy models and the proposed fuzzy structural classifier, proved to be better than using the SVM classifier with the directional features.

An approach for recognizing isolated online Arabic characters is presented in chapter 4. The novelty of the proposed approach comes from modeling the isolated Arabic online characters based on their graphemes. In the training phase, the codebook is generated from

the extracted graphemes of the training data. Then, the characters are represented based on their graphemes using the generated codebook. In the recognition phase, the graphemes of the testing data are extracted then the representations of these graphemes are combined to build the pattern of the corresponding character. Different features and different classification approaches are used in order to investigate the proposed graphemes modeling. A dataset set of isolated Arabic online characters comprising the different forms of all characters is collected to evaluate the proposed approach. Two sets of experiments were conducted. The first set was performed to evaluate the extracted graphemes and the achieved accuracy in these experiments reaches 97.02 which shows the effectiveness of the proposed process used in extracting the graphemes and building the codebook. In the second set of experiments, the testing characters are recognized using the characters' graphemes modeling of the training characters. A recognition rate of 87.53 was obtained in these experiments.

The grapheme-based approach proposed for Arabic online text recognition is presented in chapter 5. The proposed approach is based on segmenting the text into basic graphemes where each one can represent a whole character or a part of a characters' trajectory.

The grapheme models are built from pre-segmented Arabic online characters in the training phase, while Arabic online text is used in the testing phase. Therefore, the training dataset is different from the set used for testing. This illustrates the robustness of the proposed approach to handle unknown data. We choose to use a database of pre-segmented characters in the training phase instead of using isolated characters to better handle the issues related to characters' connections in an input text.

To address the problem of variability in writing, the grapheme models are built in a fuzzy manner. Moreover, the relative heights of the extracted graphemes are integrated into the graphemes' fuzzy models to address the confusion between the different graphemes that look similar.

In the recognition phase, Arabic online text is modeled as a sequence of graphemes. For this purpose, a new approach for segmenting Arabic online text into its graphemes is proposed. The algorithm utilizes the way characters are joined in Arabic online text.

A fuzzy classification approach is used to recognize the extracted graphemes from the testing data. Furthermore, the fuzzy models generated from the graphemes of the training characters are used by the fuzzy classifier to find the most similar class for each testing grapheme. A graph-based algorithm is used to map the recognized graphemes to their corresponding characters based on the graphemes' statistics gathered by analyzing the training data.

6.2 Limitations of the Work

The main objectives of this thesis are met. However, this work has some limitations that we discuss below.

- The used features are highly dependent on the order of writing. The starting point, the point from where the writer starts the writing, has a great impact on the extracted features. Different order of writing for the same sample yield different features. However, this can be solved by building models for the different writing styles.

- A semi-automated process is used to reduce the generated codebook into *bag-of-classes*. The final *bag-of-classes* are adjusted manually to ensure that each bag contains a set of classes that are similar.
- The used data sets in our experiments are either digits, isolated characters or text. The proposed approaches are not evaluated using data sets that combine digits and words.
- The used features are mainly based on the direction and length of writing. Different features can be also utilized to improve the recognition rate.

6.3 Future Research Directions

The different phases of this work can be investigated more using other options for each phase. Some directions for future research are as follows.

- **Hybrid structural/statistical classification.** Combining the strengths of structural classification with statistical classification in a hybrid structural/statistical classification is expected to perform better compared to either structural or statistical classifier. The main objective of the hybrid classification is to take the advantage of both structural and statistical classification and they are supposed to complement each other.
- **Optimizing the graphemes' codebook.** The obtained recognition rate is strongly affected by the generated graphemes' codebook. Enhancing the process of generating the codebook is expected to improve the recognition rate. The current process of generating the codebook is based on clustering the graphemes using k -

means clustering algorithm. This process may be improved by enhancing the graphemes' representation and explore different clustering algorithms.

- **Using dictionary and language model.** The obtained results can be improved by using a dictionary and language model. This can be done as a post-processing phase in which we can also utilize some natural language processing techniques to further enhance the results.

References

- [1] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 8, pp. 787–808, 1990.
- [2] R. Plamondon and S. N. Srihari, "Online and off-line handwriting recognition: a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, 2000.
- [3] C. C. Tappert and S.-H. Cha, "English language handwriting recognition interfaces." *Text entry systems: Mobility, accessibility, universality*, vol. 53, no. 9. 2007.
- [4] M. Parveze, "Arabic handwritten text recognition using structural and syntactic pattern attributes, *PhD Dissertation*," King Fahd University of Petroleum and Minerals, 2010.
- [5] "World Arabic Language Day," *UNESCO*, 2012. [Online]. Available: http://www.unesco.org/new/en/unesco/events/prizes-and-celebrations/celebrations/international-days/world-arabic-language-day/?utm_content=buffer7c141&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer.
- [6] D. C. Tran, "Bi-character model for on-line cursive handwriting recognition," *Journal of Science and Technology*, vol. 48, no. 4, pp. 1–12, 2012.
- [7] B. Al-Badar and S. A. Mahmoud, "Survey and bibliography of Arabic optical text recognition," *Signal Processing*, vol. 41, no. 1, pp. 49–77, 1995.
- [8] M. T. Parvez and S. A. Mahmoud, "Offline arabic handwritten text recognition: A survey," *ACM Computing Surveys*, vol. 45, no. 2, pp. 1–35, Feb. 2013.
- [9] M. Kherallah, A. Elbaati, H. ElAbed, and A. Alimi, "The On/Off (LMCA) Dual Arabic Handwriting Database," *11th International Conference on Frontiers in Handwriting Recognition, Quebec, Canada*, 2008.
- [10] H. El Abed, V. Märgner, M. Kherallah, and A. M. Alimi, "ICDAR 2009 Online Arabic Handwriting Recognition Competition," *10th International Conference on Document Analysis and Recognition*, pp. 1388–1392, Barcelona, Spain, 2009.
- [11] R. M. Saabni and J. A. El-Sana, "Comprehensive synthetic Arabic database for on/off-line script recognition research," *International Journal on Document Analysis and Recognition*, vol. 16, no. 3, pp. 285–294, 2013.
- [12] R. I. M. Elanwar, M. a Rashwan, and S. a Mashali, "OHASD: The First On-Line Arabic Sentence Database Handwritten on Tablet PC," *In Proceedings of World*

Academy of Science, Engineering and Technology (WASET), International Conference on Signal and Image Processing ICSIP, vol. 69, pp. 910-915. Tamil Nadu, India, 2010.

- [13] I. Abdelaziz and S. Abdou, "AltecOnDB: A Large-Vocabulary Arabic Online Handwriting Recognition Database," *arXiv Prepr. 1412.7626*, pp. 1–12, 2014.
- [14] S. Abdul Azeem, M. El Meseery, and H. Ahmed, "Online Arabic Handwritten Digits Recognition," *Frontiers in Handwriting Recognition (ICFHR) International Conference on IEEE*, pp. 135–140, bari, Italy, Sep. 2012.
- [15] N. Mezghani, A. Mitiche, and M. Cheriet, "Bayes classification of online arabic characters by Gibbs modeling of class conditional densities.," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 7, pp. 1121–31, Jul. 2008.
- [16] A. Kosmala, J. Rottland, and G. Rigoll, "Improved On-Line Handwriting Recognition Using Context Dependent Hidden Markov Models," *In Document Analysis and Recognition, Proceedings of the Fourth International Conference, IEEE*, pp. 641–644, Ulm, Germany, 1997.
- [17] M. Pastor, A. Toselli, and E. Vidal, "Writing speed normalization for on-line handwritten text recognition," *Eighth International Conference on Document Analysis and Recognition, IEEE*, pp. 1131–1135, Seoul, South Korea , 2005.
- [18] H. Eraqi and S. Abdelazeem, "An On-line Arabic Handwriting Recognition System: Based on a New On-line Graphemes Segmentation Technique," *Document Analysis and Recognition (ICDAR), International Conference*, pp. 409–413, Beijing, China, Sep. 2011.
- [19] N. Tagougui, H. Boubaker, M. Kherallah, and A. M. Alimi, "A Hybrid NN/HMM Modeling Technique for Online Arabic Handwriting Recognition," *International Journal of Computational Linguistics Research*, vol. 4, no. 3, pp. 107–118, 2013.
- [20] B. Alsallakh and H. Safadi, "AraPen: an Arabic online handwriting recognition system.," in *2nd International Conference on Information and communication technologies*, IEEE, pp. 1844– 1849, 2006.
- [21] A. T. Al-taani and S. Al-Haj, "Recognition of On-line Arabic Handwritten Characters Using Structural Features," *Journal of Pattern Recognition Research.*, vol. 1, pp. 23–37, 2010.
- [22] K. Daifallah, N. Zarka, and H. Jamous, "Recognition-Based Segmentation Algorithm for On-Line Arabic Handwriting," *10th International Conference on Document Analysis and Recognition, ICDAR'09, IEEE*, pp. 886–890, Catalonia, Spain, 2009.
- [23] S. A. Husain, A. Sajjad, and F. Anwar, "Online Urdu Character Recognition System," in *MVA2007 IAPR Conference on Machine Vision Applications*, , pp. 1–

7, Tokyo, Japan, 2007.

- [24] M. Kherallah, L. Haddad, A. M. Alimi, and A. Mitiche, "On-line handwritten digit recognition based on trajectory and velocity modeling," *Pattern Recognition. Letter*, vol. 29, no. 5, pp. 580–594, Apr. 2008.
- [25] N. Tagougui, H. Boubaker, M. Kherallah, and A. M. Alimi, "A Hybrid MLPNN / HMM Recognition System for Online Arabic Handwritten Script," in *World Congress in Computer and Information Technology (WCCIT)*, pp. 1–6, IEEE, Sousse, Tunisia, 2013.
- [26] A. Teredesai, E. Ratzlaff, J. Subrahmonia, and V. Govindaraju, "On-Line Digit Recognition using Off-Line Features," in *Proceedings of Indian Conference on Computer Vision, Graphics and Image Processing*, Ahmedabad, India 2002.
- [27] S. M. Ismail and S. N. H. S. Abdullah, "Online Arabic Handwritten Character Recognition Based on a Rule Based Approach," *Journal of Computer Science*, vol. 8, no. 11, pp. 1859–1868, Nov. 2012.
- [28] H. Boubaker, M. Kherallah, and A. M. Alimi, "New Algorithm of Straight or Curved Baseline Detection for Short Arabic Handwritten Writing," *10th International Conference on Document Analysis and Recognition, ICDAR'09, IEEE*, pp. 778–782, Catalonia, Spain, 2009.
- [29] M. I. Razzak, M. Sher, and S. A. Hussain, "Locally baseline detection for online Arabic script based languages character recognition," *International Journal of the Physical Sciences*, vol. 5, no. July, pp. 955–959, 2010.
- [30] S. A. Azeem and H. Ahmed, "Recognition of segmented online Arabic handwritten characters of the ADAB database," *10th International Conference on Machine Learning and Applications and Workshops (ICMLA)*, vol. 1, pp. 204–207, IEEE, Hawaii, USA, 2011.
- [31] J. Sternby, J. Morwing, J. Andersson, and C. Friberg, "On-line Arabic handwriting recognition with templates," *Pattern Recognition.*, vol. 42, no. 12, pp. 3278–3286, Dec. 2009.
- [32] H. Boubaker, A. Chaabouni, M. Kherallah, A. M. Alimi, and H. El Abed, "Fuzzy Segmentation and Graphemes Modeling for Online Arabic Handwriting Recognition," *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2010 pp. 695–700, Nov. Kolkata, India, 2010.
- [33] G. Kour and R. Saabne, "Real-Time Segmentation of On-Line Handwritten Arabic Script," *14th International Conference on Frontiers in Handwriting Recognition*, pp. 417–422, Crete, Greece, 2014.
- [34] M. A. Abuzaraida and A. M. Zeki, "Segmentation techniques for online Arabic handwriting recognition: a survey," in *International Conference on Information and*

Communication Technology for the Muslim World (ICT4M), IEEE, pp. D37–D40 Jakarta, Indonesia, 2010.

- [35] M. A. Abuzaraida, A. M. Zeki, and A. M. Zeki, “Online Recognition System for Handwritten Arabic Mathematical Symbols,” in *International Conference on Advanced Computer Science Applications and Technologies*, pp. 223–227, Kuala Lumpur, Malaysia, 2013.
- [36] J. M. Tan, “Automatic verification of the outputs of multiple classifiers for unconstrained handwritten numerals,” *PhD Dissertation*, Concordia University, 2004.
- [37] Cheng-Lin Liu, S. Jaeger, and M. Nakagawa, “Online recognition of chinese characters: the state-of-the-art,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 198–213, Feb. 2004.
- [38] V. Lorigo, L. and Govindaraju, “Offline Arabic Handwriting Recognition: A Survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, pp. 712–724, 2006.
- [39] H. Almuallim and S. Yamaguchi, “A Method of Recognition of Arabic Cursive Handwriting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 224, no. 5, pp. 715–722., 1987.
- [40] S. Al-Emami and M. Usher, “On-Line Recognition of Handwritten Arabic Characters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 704–710, 1990.
- [41] H. S. M. Beigi, K. Nathan, G. J. Clary, and J. Subrahmonia, “Challenges of Handwriting Recognition in Farsi , Arabic and Other Languages with Similar Writing Styles An On-line Digit Recognizer,” *Proceedings of the 2nd Annual Conference on Technological Advancements in Developing Countries*, Columbia University, New York. 1994.
- [42] M. Kherallah, S. Njah, A. M. Alimi, and N. Derbel, “Recognition of on-line handwritten digits by neural networks using circular and beta approaches,” *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 164–169, Tunisia, 2002.
- [43] M. Kherallah, A. M. Alimi, and N. Derbel, “On-Line Recognition of Handwritten digits by Self Organisation Maps Using Elliptical and Beta Representations,” *In Proceedings of the IEEE International Conference on SCS’04*, pp. 503–507, Mounastir, Tunisia, 2004.
- [44] A. Al-taani and M. Hammad, “Recognition of On-line Handwritten Arabic Digits Using Structural Features and Transition Network,” *Informatica*, vol. 32, pp. 275–281, 2008.

- [45] S. Izadi and C. Y. Suen, "Online Writer-Independent Character Recognition Using a Novel Relational Context Representation," *Seventh International Conference on Machine Learning and Applications*, pp. 867–870, California, USA, 2008.
- [46] N. Mezghani, A. Mitiche, and M. Cheriet, "On-line recognition of handwritten Arabic characters using A Kohonen neural network," in *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop IEEE*, pp. 490–495. Ontario, Canada, 2002.
- [47] N. Mezghani, A. Mitiche, and M. Cheriet, "Combination of pruned Kohonen maps for on-line arabic characters recognition," in *Seventh International Conference on Document Analysis and Recognition, ICDAR*, pp. 900–904, Edinburgh, Scotland 2003.
- [48] S. C. Zhu, Y. N. Wu, and D. Mumford, "Minimax entropy principle and its application to texture modeling," *Neural computation*, vol. 9, no. 8, pp. 1627–1660, 1997.
- [49] A. T. Al-taani, "An Efficient Feature Extraction Algorithm for the Recognition of Handwritten Arabic Digits," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 2, pp. 06 –23, 2008.
- [50] M. A. H. Omer and S. L. Ma, "Online Arabic Handwriting Character Recognition Using Matching Algorithm," in *The 2nd International Conference on Computer and Automation Engineering (ICCAE), Vol. 2, IEEE.*, pp. 259–262, Singapore, Singapore, 2010.
- [51] I. Khodadad, M. Sid-ahmed, and E. Abdel-raheem, "Online Arabicpersian Character Recognition Using Neural Network," In *54th International Midwest Symposium on Circuits and Systems (MWSCAS), IEEE*, Seoul, Korea, 2011.
- [52] G. Kour and R. Saabne, "Fast classification of handwritten on-line Arabic characters," in *6th International Conference of Soft Computing and Pattern Recognition (SoCPaR), IEEE*, pp. pp. 312–318, Tunisia, 2014.
- [53] M. Kherallah, F. Bouri, and A. M. Alimi, "On-line Arabic handwriting recognition system based on visual encoding and genetic algorithm," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 1, pp. 153–170, Feb. 2009.
- [54] S. Abdelazem and H. M. Eraqi, "On-line Arabic Handwritten Personal Names Recognition System Based on HMM," in *2011 International Conference on Document Analysis and Recognition*, pp. 1304–1308, Beijing, China, 2011.
- [55] R. I. Elanwar, M. Rashwan, and S. Mashali, "Unconstrained Arabic Online Handwritten Words Segmentation using New HMM State Design," in *Proceedings of World Academy of Science, Engineering and Technology.*, vol. 6, pp. 1189–1197, 2012.

- [56] G. Al-Habian and K. Assaleh, "Online Arabic handwriting recognition using continuous Gaussian mixture HMMS," *International Conference on Intelligent and Advanced Systems*, pp. 1183–1186, Kuala Lumpur, Malaysia, 2007.
- [57] H. A. A. Alshafy and M. E. Mustafa, "HMM Based Approach for Online Arabic Handwriting recognition," in *14th International Conference on Intelligent Systems Design and Applications (ISDA), IEEE.*, , pp. 211–215, Okinawa, Japan, 2014.
- [58] F. Faradji, K. Faez, and M. S. Nosrati, "Online farsi handwritten words recognition using a combination of 3 cascaded RBF neural networks," *International Conference on Intelligent and Advanced Systems, ICIAS 2007*, pp. 134–138, Kuala Lumpur, Malaysia, 2007.
- [59] K. Assaleh, T. Shanableh, and H. Hajjaj, "Recognition of handwritten Arabic alphabet via hand motion tracking," *Journal of the Franklin Institute*, vol. 346, no. 2, pp. 175–189, 2009.
- [60] R. I. Elanwar, M. A. Rashwan, and S. A. Mashali, "Simultaneous Segmentation and Recognition of Arabic Characters in an Unconstrained On-Line Cursive Handwritten Document," in *Proceedings of world academy of science, engineering and technology*, vol. 23, pp. 288–291, 2007.
- [61] M. Abuzaraida, A. Zeki, and A. Zeki, "Online Recognition System for Handwritten Arabic Digits," in *The 7th International Conference on Information Technology*, , pp. 45–49, Amman, Jordan, May 2015.
- [62] R. Saabni and J. El-Sana, "Hierarchical on-line Arabic handwriting recognition," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 867–871, Catalonia, Spain, 2009.
- [63] H. Boubaker, M. Kherallah, and A. Alimi, "New Strategy for the On-Line Handwriting Modelling," *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, pp. 1233–1247, Curitiba, Brazil, 2007.
- [64] J. Sternby, "An Additive On-line Single Character Recognition Method," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, pp. p.417–422, La Baule, France, 2006.
- [65] J. Sternby and C. Friberg, "The recognition graph - Language independent adaptable on-line cursive script recognition," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 14–18, Seoul, Korea, 2005.
- [66] H. Li and L. Yang, "Extensions and relationships of some existing lower-bound functions for dynamic time warping," *Journal of Intelligent Information Systems*, vol. 43, no. 1, pp. 59–79, 2014.
- [67] S. G. Salve and K. Jondhale, "Shape Matching and Object Recognition Using Shape Contexts," *International Conference on Computer Science and Information*

Technology, pp. 471–474, Chengdu, China, 2010.

- [68] R. Saabni, “The Multi Angular Descriptor (MAD): a Binary and Gray Images Descriptor for Shape Recognition,” in *In Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing, ACM.*, pp. 53–58, 2013.
- [69] B. Bataineh, S. N. H. S. Abdullah, and K. Omar, “A Statistical Global Feature Extraction Method for Optical Font Recognition,” in *Intelligent Information and Database Systems. Springer Berlin Heidelberg*, vol. 1, pp. 257–267, 2011.
- [70] S. Impedovo, F. M. Mangini, and D. Barbuizi, “A novel prototype generation technique for handwriting digit recognition,” *Pattern Recognition.*, vol. 47, no. 3, pp. 1002–1010, 2014.
- [71] P. P. S. Subhashini and V. V. K. D. V Prasad, “Recognition of Handwritten Digits Using Rbf Neural Network,” *International Journal of Research in Engineering and Technology*, pp. 393–397, 2013.
- [72] X. X. Niu and C. Y. Suen, “A novel hybrid CNN-SVM classifier for recognizing handwritten digits,” *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, 2012.
- [73] W.-L. Jiang, Z.-X. Sun, B. Yuan, W.-T. Zheng, and W.-H. Xu, “User-Independent Online Handwritten Digit Recognition,” in *Machine Learning and Cybernetics, 2006 IEEE international Conference*, pp. 3359–3364, Dalian, China, August 2006.
- [74] J. H. Alkhateeb and M. Alseid, “DBN - Based learning for Arabic handwritten digit recognition using DCT features,” in *6th International Conference on Computer Science and Information Technology, CSIT 2014 - Proceedings*, pp. 222–226, Amman, Jordan, 2014.
- [75] J. Sadri, C. Y. Suen, and T. D. Bui, “Application of Support Vector Machines for Recognition of Handwritten Arabic/Persian Digits,” *Proceedings of Second Iranian Conference on Machine Vision and Image Processing*, vol. 1, pp. 300–307, 2003.
- [76] S. Mozaffari and K. Faez, “Structural Decomposition and Statistical Description of Farsi / Arabic Arabic handwritten numeric characters,” in *Proceedings of Eighth International Conference on Document Analysis and Recognition, IEEE*, pp. 237–241, Seoul, Korea, 2005.
- [77] S. A. Mahmoud, “Arabic (Indian) handwritten digits recognition using Gabor-based features,” in *International Conference on Innovations in Information Technology*, Al Ain, United Arab Emirates. pp. 683–687, Dec. 2008.
- [78] S. A. Mahmoud, “Recognition of writer-independent off-line handwritten Arabic (Indian) numerals using hidden Markov models,” *Signal Processing*, vol. 88, no. 4, pp. 844–857, Apr. 2008.

- [79] S. A. Mahmoud and W. G. Al-Khatib, "Recognition of Arabic (Indian) bank check digits using log-gabor filters," *Applied Intelligence*, vol. 35, no. 3, pp. 445–456, May 2010.
- [80] S. Abdleazeem and E. El-Sherif, "Arabic handwritten digit recognition," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 11, no. 3, pp. 127–141, Nov. 2008.
- [81] I. Abuhaiba, S. Mahmoud, and R. J. Gree, "Recognition of handwritten cursive Arabic characters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 664–672, Jun. 1994.
- [82] M. T. Parvez and S. A. Mahmoud, "Arabic handwriting recognition using structural and syntactic pattern attributes," *Pattern Recognition*, vol. 46, no. 1, pp. 141–154, Jan. 2013.
- [83] K. Halawani, "Recognition of Arabic Online Handwritten Text Using Syntactical Techniques," *MSc Thesis*, King Fahd University of Petroleum and Minerals, 2013.
- [84] N. Tagougui, M. Kherallah, and A. M. Alimi, "Online Arabic handwriting recognition: A survey," *International Journal on Document Analysis and Recognition*, vol. 16, no. 3, pp. 209–226, 2013.
- [85] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature.," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.
- [86] M. A. Abuzaraida, A. M. Zeki, and A. M. Zeki, "Feature Extraction Techniques of Online Handwriting Arabic Text Recognition," in *5th International Conference on Information and Communication Technology for the Muslim World.*, Rabat, Morocco, 2013.
- [87] H. Xu, "A Support System for Graphics for Visually Impaired People," *MSc. Thesis*, The University of Western Ontario, 2013.
- [88] K. Jung and H. J. Kim, "On-line recognition of cursive Korean characters using graph representation," *Pattern Recognition.*, vol. 33, pp. 399–412, 2000.
- [89] H. Boubaker, A. Chaabouni, M. Ben Halima, A. El Baati, and H. El Abed, "Arabic diacritics detection and fuzzy representation for segmented handwriting graphemes modeling," *2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, pp. 71–76, Aug. Tunisia, 2014.
- [90] H. Boubaker, N. Tagougui, H. El Abed, M. Kherallah, and A. M. Alimi, "Graphemes Segmentation for Arabic Online Handwriting Modeling," *Journal of information processing systems*, vol. 10, no. 4, pp. 1–20, 2014.

- [91] I. Abdelaziz, S. Abdou, and H. Al-barhamtoshy, "Large Vocabulary Arabic Online Handwriting Recognition System," *arXiv Prepr. 1410-4688*, 2014.
- [92] G. Kour, "Real-time segmentation and recognition of on-line handwritten Arabic script," *MSc. Theses*, Tel Aviv University, 2015.
- [93] H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," *IRE Transactions on Electronic Computers EC*, vol. 2, pp. 260–268, 1961.
- [94] K. Addakiri and M. Bahaj, "On-line Handwritten Arabic Character Recognition using Artificial Neural Network," *International Journal of Computer Applications*, vol. 55, no. 13, pp. 42–47, 2012.
- [95] B. Alijla and K. Kwaik, "OIAHCR : Online Isolated Arabic Handwritten Character Recognition Using Neural Network," *The International Arab Journal Of Information Technology*, vol. 9, no. 4, pp. 343–351, 2012.
- [96] J. C. Léger, "Menger curvature and rectifiability," *Annals of Mathematics*, vol. 149, no. 3, pp. 831–869, 1999.
- [97] I. Guyon, P. Albrecht, Y. L. E. Cun, J. Denkert, W. Hubbardt, and T. B. Laboratories, "Design of A Neural Network Character Recognizer for a Touch Terminal," *Pattern Recognition.*, vol. 24, pp. 105–119, 1991.
- [98] A. Amin, "Off-line Arabic character recognition: The state of the art," *Pattern Recognition.*, vol. 31, no. 5, pp. 517–530, 1998.
- [99] A. Al-Shatnawi and K. Omar, "Methods of Arabic Language Baseline Detection – The State of Art," *Journal of Computer Science and Network Security*, vol. 8, no. 10, pp. 137–143, 2008.
- [100] M. Pechwitz and V. Margner, "Baseline estimation for Arabic handwritten words," in *In Frontiers in Handwriting Recognition, Proceedings. Eighth International Workshop . IEEE*, pp. 479–484, Ontario, Canada, 2002.
- [101] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, "The HTK Book (v3. 4), " 2006.
- [102] S. A. Mahmoud, I. Ahmad, W. G. Al-Khatib, M. Alshayeb, M. Tanvir Parvez, V. Märgner, and G. A. Fink, "KHATT: An open Arabic offline handwritten text database," *Pattern Recognition.*, vol. 47, no. 3, pp. 1096–1112, 2014.
- [103] A. M. Alimi, "An Evolutionary Neuro-Fuzzy Approach to Recognize On-Line Arabic Handwriting," in the *Proceedings of the Fourth International Conference on Document Analysis and Recognition, IEEE*, pp. 382–386, Ulm, Germany, 1997.

Vitae

Name: Dhiaa Abdulrab Ali Musleh

Nationality: Yemeni

Date of Birth: 1/1/1976

Email: dhia@hotmail.com

Address: Yemen, Taiz, Gamal Street

Academic Background:

Ph.D. in Computer Science and Engineering, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, 2016.

M.Sc. in Computer Science, King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, 2010.

B.S. in Computer Science, University of Mosul, Mosul, Iraq, 2000.

PUBLICATIONS

Following is the list of publications, at the time of submitting this dissertation, based on the work in this dissertation

Journal Paper

Dhiaa A. Musleh, Khaldoun M. Halawani and Sabri A. Mahmoud, “Fuzzy Modeling for Handwritten Arabic Numeral Recognition”, accepted for publication in the International Arab Journal of Information Technology (ISI).

Patent

Dhiaa A. Musleh, Khaldoun M. Halawani and Sabri A. Mahmoud, “An Arabic Like Alphanumeric Character Recognition System and Method Using Automatic Fuzzy Modeling”, (Patent Pending), 2016, U.S. Patent and Trademark Office (USPTO).